AI ADOPTION IN SOFTWARE DEVELOPMENT TEAMS

TUBA BALTA

BOĞAZİÇİ UNIVERSITY

2025

AI ADOPTION IN SOFTWARE DEVELOPMENT TEAMS

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Management Information Systems

by

Tuba Balta

Boğaziçi University

2025

AI Adoption in Software Development Teams

The thesis of Tuba Balta has been approved by:

Assoc. Prof. Nazım Taşkın _____
(Thesis Advisor)

Assoc. Prof. Nazım Ziya Perdahçı _____
(Thesis Advisor)

Assoc. Prof. Murat Levent Demircan _____
(External Member)

June 2025

DECLARATION OF ORIGINALITY

I, Tuba Balta, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all sources of ideas and words, including digital resources, which have been produced or published by another person or institution;
- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- this is a true copy of the thesis approved by my advisor and thesis committee at Boğaziçi University, including final revisions required by them.

Signature……...

Date…….....

ABSTRACT

AI Adoption in Software Development Teams

This thesis investigates the factors affecting the adoption of Artificial Intelligence, with focus on Large Language Models (LLMs), among software professionals. Previous technology adoption literature is examined, and a novel adoption model is suggested based on Unified Theory of Acceptance and Use of Technology (UTAUT) and Protection Motivation Theory (PMT). The model aimed to explain the adoption factors by considering both benefit and risk perceptions associated with LLMs use. The model is tested using survey data collected from 151 respondents and examined using Partial Least Squares Structural Equation Modelling (PLS-SEM). From a theoretical perspective, this study contributes to existing technology adoption literature by extending with PMT integration, offering a systematic approach to incorporate risk perception into LLM adoption. The results reveal that adoption decisions are predominantly benefit-driven, as performance expectancy, effort expectancy, and task efficiency positively influence adoption intention. On the other hand, PMT constructs did not exhibit a direct effect on adoption intention, suggesting that risk perceptions do not yet play a decisive role in Large Language Model adoption among software professionals.

ÖZET

Yazılım Geliştirme Ekiplerinde Yapay Zekanın Benimsenmesi

Bu tez, yazılım geliştirme ekiplerinde, Yapay Zekanın, özellikle Büyük Dil Modellerinin (BDM), benimsenmesini etkileyen faktörleri araştırmaktadır. Mevcut teknoloji benimseme literatürü incelenmiş ve Teknolojinin Kabulü ve Kullanımı için Birleştirilmiş Teori (UTAUT) ile Koruma Motivasyonu Teorisi'ne (PMT) dayanan özgün bir benimseme modeli önerilmiştir. Bu model, BDM kullanımına ilişkin fayda ve risk algılarını birlikte ele alarak benimseme faktörlerini açıklamayı amaçlamaktadır. Model, 151 katılımcıdan toplanan anket verileriyle test edilmiş ve Kısmi En Küçük Kareler Yapısal Eşitlik Modellemesi (PLS-SEM) yöntemi ile analiz edilmiştir. Kuramsal açıdan bakıldığında, bu çalışma mevcut teknoloji benimseme literatürüne PMT entegrasyonu ile katkı sağlamakta; BDM benimsenmesine risk algısını sistematik biçimde dahil eden bir yaklaşım sunmaktadır. Bulgular, benimseme kararlarının büyük ölçüde fayda odaklı olduğunu göstermektedir; performans beklentisi, çaba beklentisi ve görev verimliliği benimseme niyeti üzerinde pozitif etki yaratmıştır. Buna karşın, PMT yapıları benimseme niyeti üzerinde doğrudan bir etki göstermemiştir; bu da, yazılım profesyonelleri arasında Büyük Dil Modeli benimsenmesinde risk algılarının henüz belirleyici bir rol oynamadığını ortaya koymaktadır.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Artificial intelligence (AI) popularity is increasing since late 2022, when advanced text generation tools began receiving widespread attention when OpenAI pioneered their chatbot. It has entered sectors such as healthcare (Bekbolatova et al., 2024), education (Adiguzel et al., 2023), economy (Gonzales, 2023) and software development (Chen et al., 2021). These technologies have not only used for personal experiments, but they have been integrated into professional workflows. One significant advancement is the rise of Large Language Models (LLMs). Such models use Deep Learning algorithms to generate human-like text (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023). LLMs are a milestone in AI history with their use of the Transformer architecture which enables them to process and produce long text with context-aware precision and fluency (Vaswani et al., 2017).

Software teams increasingly use LLMs for various tasks and these models are becoming an integral part of the software development workflow. Their applications span the entire development lifecycle, from initial requirement gathering through code creation to troubleshooting and ongoing maintenance activities (Chen et al., 2021; Fan et al., 2023). Some researchers revealed that their adoption leads to higher productivity, reduced error rates and increased quality through automation of routine coding tasks and assisted problem solving (Fan et al., 2023).

However, the widespread integration of AI technologies into existing software development processes also poses challenges such as hallucinations, legal uncertainties or ethical concerns that could impact both the effectiveness and acceptability of such technologies in professional settings (Huang et al., 2025).

At the same time, using LLMs is not always straightforward. One concern is that these models can sometimes give answers that seem accurate but are wrong. They might generate code that looks plausible but contains logic errors or security issues that are easy to miss (Mohsin et al., 2024). These kinds of mistakes, often called "hallucinations", can be especially risky in situations where the software must be reliable and precise in domains such as healthcare (Huang et al., 2025).

Another important issue is related to legal risks. Since LLMs are trained on large volumes of data, they can end up reproducing copyrighted content violating intellectual property laws (Liu et al., 2024). They can expose personal information (Borji, 2023; Carlini et al., 2021) or sensitive business data (Carlini et al., 2021; Khan et al., 2024) violating data protection regulations like the General Data Protection Regulation (GDPR) (Feretzakis et al., 2025). Many studies highlights that LLMs generate biased output (Borji, 2023; Dastin, 2022; Fuster et al., 2022; Garcia et al., 2024; Vig et al., 2020). This bias stems from the training data which can result in unfair outputs (Fuster et al., 2022). Beside these concerns, there is confusion about who is responsible when problems arise from content generated by an AI system (Novelli et al., 2024).

Various studies have identified and also formulated risk mitigation strategies. There are different strategies that can be employed at the organizational or individual level. For instance, Wang and Chen (2023) suggest that rigorous review processes by human developers can mitigate legal and security risks. Varghese and Thomas (2024) suggest training developers on the limitations and security pitfalls of using AI-generated code. The European Union's AI Act has established strict requirements for human oversight in AI systems deployment. This emphasis on human validation reflects a growing consensus among regulators and researchers. Despite the

capabilities of LLMs to generate code, human expertise remains essential for verifying quality, security and compliance aspects of AI-assisted software development.

Understanding software professionals' responses to emerging LLM technology requires examining both risk perception and benefits through established theoretical frameworks. In the Management Information System (MIS) literature, the Protection Motivation Theory by Rogers (1975) provides insights for this study's investigation of LLM risks. PMT explains how individuals evaluate risks and adopt protective behaviors based on their perception of vulnerability, severity, response efficacy and self-efficacy. In the context of LLM usage, PMT helps explain how software professionals assess potential risks in their development processes and subsequently implement mitigation strategies. On the other hand, LLMs have also brought immense amount of benefits to software development through automated or assisted code optimizations (Shypula et al., 2024), code generation (Jiang et al., 2024), debugging (Tian et al., 2024) or documentation (Dvivedi et al., 2024; J. Y. Khan & Uddin, 2023). Given these applications, benefit factors cannot be ignored when studying LLM adoption. Therefore, benefit factors from the Unified Theory of Acceptance and Use of Technology (UTAUT) by Venkatesh et al. (2003) were employed, which will be used to study the performance expectancy, effort expectancy and task efficiency that drive technology adoption of LLMs. With these theoretical underpinnings, this study has introduced a new conceptual model that accounts for both the protective motivations arising from perceived risks and the utilitarian motivations arising from perceived benefits of LLM integration in software development.

In essence, the current literature provides pieces of the puzzle. There are studies about adoption trends (Weisz et al., 2024), anecdotal developer concerns (Kuhail et al., 2024) and evidence of technical risks (Pearce et al., 2022) – but to our knowledge, no prior work has yet developed a conceptual model integrating PMT (Rogers, 1975) and UTAUT (Venkatesh et al., 2003) to understand the risk perceptions of software professionals compared to the benefits of a new technology. To this end, it is aimed to fill this gap in the literature with a questionnaire of software professionals about their risks perception and adoption of LLMs and study the results with a custom model based on PMT and UTAUT.

The literature review part begins by introducing background knowledge for the Software Development Life Cycle (SDLC), outlining its phases and different applications of it over time. Next, historical perspective on LLMs starting from their development from statistical models to current Transformer-based architectures provided. The review will then examine the intersection points between these two fields, highlighting how LLMs are being integrated into various SDLC phases, from requirements gathering to maintenance. After this, the risks associated with LLM outputs in software development will be analyzed. The literature review will conclude by exploring emerging mitigation strategies for these risks.

To understand software professionals' perspectives on LLM adoption, a quantitative survey conducted based on the conceptual model combining PMT and UTAUT. Partial Least Squares Structural Equation Modeling (PLS-SEM) is employed to analyze the relationships between perceived risks, benefits, and adoption behavior.

This study aims to fill in the gap for a quantitative study for the adoption of LLMs in software teams including risk factors of LLMs. To this end, this study aims to answer following questions:

- What are the factors that influence software professionals' decision to adopt LLM in their work?
- To which extent do the factors matter?

This thesis is organized as follows: This chapter introduced the context of the study, outlined the theoretical background, and presented the research objectives. Chapter 2 reviews the literature on the Software Development Life Cycle, the emergence and capabilities of Large Language Models, their integration into software development life cycle, and the associated risks and mitigation strategies. Chapter 3 introduces the theoretical model developed through the integration of Protection Motivation Theory and the Unified Theory of Acceptance and Use of Technology and presents the hypothesis. In the following Chapter 4, the research design is outlined, including details of the survey instrument, data collection procedures, and analysis method. In Chapter 5, the analysis results are presented. Finally, Chapter 6 concludes the thesis with a discussion of the findings, implications, limitations and future research directions.

CHAPTER 2

LITERATURE REVIEW

2.1 Software development life cycle (SDLC)

The Software Development Life Cycle (SDLC) is a structured process for creating software systems from inception to completion. SDLC defines the entire procedures of software development step by step. Various software development methodologies have emerged to implement SDLC in projects (Sommerville, 2016).

Traditional SDLC methodologies include the Waterfall methodology, iterative methodology, spiral methodology, V-model, and Agile (Balaji & Murugaiyan, 2012). Each of these methodologies differs in their way of managing the development process. Their applicability depends on project requirements, team dynamics, and organizational priorities.

Understanding the SDLC phases could increase team productivity by adjusting the development paradigm better to the project requirements. Each methodology influences the responsibilities of team members and how they collaborate in different ways. In the Waterfall methodology, tasks are divided into distinct phases with clear handoffs between roles in a goal- and plan-oriented manner (Balaji & Murugaiyan, 2012). For instance, business analysts focus on requirement gathering, which is followed by designers creating the system architecture, then developers implementing the solution, and finally testers ensuring quality before deployment. Conversely, in the Agile methodology, roles are more fluid and collaborative. Teams include a Scrum Master for processes, product owners who continuously refine requirements based on stakeholder feedback and the remaining

roles named developers who have done the development and testing (Alsaqqa et al., 2020).

These differences in team roles are related to the structure of SDLC itself. SDLC consists of analysis, design, implementation, testing, deployment, and maintenance phases but can adapt based on the chosen methodology (Alsaqqa et al., 2020). For instance, Scrum, an agile framework, includes three main phases in its lifecycle, namely outline planning or the pre-sprint phase, the development or the sprint phase, and the project closure phase (Alsaqqa et al., 2020).

Roles and responsibilities in software development further reflect these methodological differences, starting from the first phase — analysis. This phase involves activities like gathering stakeholders' needs and expectations and elicitation of these requirements. It typically falls under the responsibilities of business analysts (BABOK, n.d.). The role plays a critical part in requirements elicitation to document the necessary information.

The next phase is the design phase. In this phase, it is expected to specify the software components and the entire system architecture. It includes selecting appropriate technologies and designing databases and user interfaces. These designs are typically led by software architects. The purpose is to ensure that the design aligns with the project requirements while prioritizing systems' maintainability and scalability (Sommerville, 2016).

After the design phase, the next step is development, also known as the implementation phase. This phase involves translating design documents into executable code. During this phase, developers are the responsible for writing code, performing code tests, and integrating components. Their primary goal is to create a functional software that meets to the project specifications (Sommerville, 2016).

The next phase is testing. It is initially carried out by programmers who identify and resolve errors themselves, ensuring the software behaves as intended. Following this, the quality assurance team takes over, and verifies functionality and performance (Naik & Tripathy, 2011). This exhaustive testing prevents issues during production deployments.

Lastly, the maintenance phase focuses on updates on the system and supporting users. Maintenance phase, also known as post-deployment, responsible for keeping the software functional, maintaining its security, and supporting the evolving stakeholder requirements. It helps maintain code quality (Gokarna & Singh, 2021). Maintenance activities can be corrective such as addressing bugs, adaptive such as transforming to new technologies, perfective such as enhancing performance, and preventative such as preventing future problems. Dedicated maintenance teams manage these tasks. DevOps is also part of the maintenance phase and introduces continuous integration (CI) practices, allowing developers to frequently merge code changes into a shared repository while ensuring that these changes pass all tests (Gokarna & Singh, 2021).

2.2 Large language models (LLMs)

Artificial Intelligence (AI) is a computer science discipline of creating something that does not depend on preprograming all problem-solving behaviour (Fogel, 2022). It had a profound impact on various fields including the economy (Gonzales, 2023), society (Gruetzemacher & Whittlestone, 2019) and education (Adiguzel et al., 2023). One notable advancement has been development of systems capable of understanding and generating human-like text.

LLMs are trained on extensive datasets and generate human-like text (Brown et al., 2020). Recent models are using the Transformer architecture, introduced by (Vaswani et al., 2017), which processes text by understanding the relationships between all the words in a sentence.

2.2.1 History

The journey of language models began with the need to enable computers to understand and generate human language (Brown et al., 2020). Early studies (Brown et al., 1992; Jelinek, 1998) focused on statistical models for predicting word sequences, such as n-grams, which predict the likelihood of a word based on the previous few words. While effective for simple tasks, these models struggled with understanding longer sentences and context which limit their utility.

Following the limitations of statistical models, neural language models (NLMs) are the next milestone in language modelling. By using neural networks, they address the shortcomings of their predecessor models. These models represent words as continuous vectors in a high-dimensional space which capture semantic relationships (Bengio et al., 2000). Back in the early days of neural network research, feedforward models were the foundations of more advanced architectures. However, they processed information in a single direction and could not really handle broader context (Bengio et al., 2000). This limitation eventually led researchers to develop Recurrent Neural Networks (RNNs). Researchers introduced these networks to keep earlier inputs in memory. What made RNNs valuable was their ability to preserve a running internal state, described as a “hidden state”, which helped link new inputs to previous ones. This made RNNs useful in tasks like language modelling, where understanding the broader sentence is often necessary for interpreting each word

correctly. That said, these models were not without limitations. One problem was the vanishing gradient problem, which made it hard for them to learn long-range dependencies between words effectively.

To address this limitation, Long Short-Term Memory (LSTM) networks is introduced, which extend the capabilities of RNNs. LSTMs are differentiated from RNNs by using internal gates mechanisms that regulate the flow of information by determining what to input, forget, and output. These input, forget, and output gates, allowing the model to operate with greater selectivity. As a result, LSTMs are more effective in modelling long-range dependencies in data and help to mitigate issues RNNs have faced such as vanishing gradients (Hochreiter & Schmidhuber, 1997). An example of LSTMs application is machine translation, where LSTMs can align meanings across languages, even when related terms are separated by considerable distance within a sentence. Despite their advantages, LSTMs are computationally expensive and struggle with very long sequences.

Various machine learning algorithms have played roles in different aspects of language processing alongside RNN and LSTM. Within the supervised learning paradigm, traditional algorithms, such as Naive Bayes and Support Vector Machines (SVMs) were effective for specific tasks like text classification, spam detection and sentiment analysis (Goswami et al., 2020). In the unsupervised learning methods, such as K-means clustering were used on unlabeled data. They were applied to large datasets to cluster them and reduce their dimensionality to reveal their hidden structures (Hastie et al., 2009). While these earlier models improved upon earlier statistical models, they struggled with long-term dependencies and context, highlighting the need for more sophisticated approaches like networks with more

layers and more sophisticated architectures, such as Transformer (Vaswani et al., 2017).

Transformers, introduced by (Vaswani et al., 2017), unlike previous models, used a mechanism called self-attention. This mechanism allows them to consider all words in a sentence at once rather than processing them sequentially. This innovation enabled the models to handle larger datasets as well as modelling the relationships between words. When exposed to larger datasets, these models better learn the statistical properties of language, including grammar, syntax, semantic and contextual relationships between words, phrases, and paragraphs. This increased language understanding in turn allows the models to better understand language and generate coherent texts.

Transformers became the foundation for today's LLMs, such as BERT (Devlin et al., 2018) and GPT series (Brown et al., 2020). They perform well in different applications from chatbots to text summarization. While BERT established transfer learning for understanding tasks, powering applications like semantic search and text classification, GPT focused on generative capabilities, enabling machines to produce contextually appropriate text. Both models influenced most of the subsequent LLMs.

Modern language models are trained on massive amounts of data and contain billions of parameters that allow the models to perform code generation and writing auto-completion, grammar assistance, game narrative generation, improving search engine responses, and answer questions with minimal or no additional training (Brown et al., 2020). These models show the importance of combining scale, data and continuous architecture improvements.

As LLMs continue to evolve, their potential to transform industries and enhance human-computer interaction grows, but so do concerns about ethical use and biases (Bender et al., 2021), which are yet to be fully addressed and thus remain an active research area.

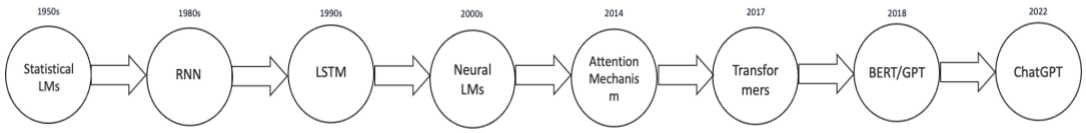

Figure 1. A brief history of language model

2.2.2 Capabilities and features of LLMs

LLMs exhibit exceptional natural language generation abilities. These capabilities enable them to engage in a wide range of complex tasks. For instance, they can translate languages with high accuracy. Text summarization is another strength, allowing LLMs to condense long documents while preserving key information. They excel at question answering by retrieving relevant information and formulating coherent responses. Creative applications include writing poetry, fiction, and generating high-quality blog posts. LLMs can maintain contextual awareness in conversations, responding appropriately to various prompts. They can explain complex subjects, concepts, and themes in accessible ways. Additionally, these models can analyze existing code to fix errors or generate entirely new code based on specifications (Adiguzel et al., 2023).

Their architecture allows them to maintain context over long sequences of text, understand nuanced meanings, and generate contextually appropriate outputs. A key feature of LLMs is their adaptability through zero-shot and few-shot learning capabilities. Zero-shot learning means the model's ability to perform tasks without any specific training examples. They rely on task descriptions or instructions. Few-

shot learning enables models to learn new tasks with a minimum number of examples. Typically ranging from one to a few dozen samples (Brown et al., 2020). These capabilities are enhanced more with prompt engineering techniques, such as Chain-of-Thought prompting, which guides the model through step-by-step reasoning processes (Li et al., 2024).

2.2.3 Training of LLMs

Pre-training and Fine-tuning

Large Language Models are trained in two stages which are pre-training and fine-tuning. Pre-training allows the model to learn language structures and semantics by training on large, diverse datasets. Fine-tuning enables the LLM to specialize in a specific task. With this approach, models can acquire the ability to solve domain-specific tasks (Brown et al., 2020; Devlin et al., 2018).

In the pre-training phase, LLMs are exposed to a wide range of text such as books, articles, and websites. In this process, training is unsupervised because it does not require labeled data. Instead, the model learns patterns by predicting the next token from data samples (Liu et al., 2024).

After pre-training, LLMs are fine-tuned to adapt their language understanding to domain-specific tasks. In this stage, it involves supervised learning. The model is trained on a curated dataset containing examples relevant to the target task (Liu et al., 2024). Fine-tuning for software development domain might include code snippets, programming documentation, or problem descriptions. Fine-tuning increases the model's ability to produce syntactically and semantically more accurate code. The final models can be used for tasks like code generation, debugging, and code completion. Research by Chen et al. (2021) demonstrated that fine-tuning LLMs on

code repositories significantly improves their performance in code understanding and generation which makes them more useful for software developers.

2.3 Applications of LLMs in software development

LLMs have transformed various aspects of software development through their diverse applications (Chen et al., 2021). Software development tasks vary from routine code maintenance to designing system architecture. Each activity requires different levels of time investment and expertise. By integrating LLMs into these activities, development teams can receive intelligent assistance across this spectrum of tasks. This integration accelerates development cycles and potentially improves code quality by reducing common programming errors and suggesting more sophisticated solutions that might not be immediately apparent to developers (Wadhwa et al., 2023).

These benefits materialize across a wide range of concrete software development applications. LLMs have been successfully deployed for tasks such as code completion, code translation, code repair, and code summarization (Hou et al., 2024). Fan et al. (2023) notes that when LLMs are fully integrated into the software lifecycle they will drive coding, design, requirements, repair, refactoring, performance improvement, documentation, and analytics.

2.3.1 Code generation and improvement

A particularly relevant application of LLMs is code generation, which aligns closely with software developers' core responsibilities. Code generation is the transformation of natural language descriptions into corresponding code implementations (Jiang et al., 2024). LLMs have shown to be capable of producing

sound code- syntactically correct code, follows programming language rules and executes without errors to achieve its intended functionality. While LLMs can generate boilerplate code, they are also capable of generating common code patterns for several programming languages. LLMs can solve complex algorithmic challenges, such as tools like GitHub Copilot, powered by OpenAI's Codex leverage pre-training on extensive datasets containing natural language and source code to produce outputs tailored to user-defined specifications (Chen et al., 2021).

LLMs are trained on diverse datasets (Fan et al., 2023) to understand the syntax and semantics of programming languages, along with a variety of common patterns. This comprehensive training enables them to develop an understanding of code structure and relationships between different programming elements. Building upon this foundation, LLMs can analyze both immediate context and overall program structure. For example, codex and codellama can determine the function of identifiers in code. This allows the model to understand the behaviour of code and detect logical flaws, such as incorrect checks, then recommend corrections based on the intended behavior (Lee et al., 2024).

Expanding the potential applications, LLMs can optimize code with regards to algorithmic complexity or runtime performance. Shypula et al. (2024) introduce a framework that adopts LLMs for high-level code optimizations. The authors trained an LLM to suggest performance-optimizing code edits. Their results demonstrate that LLMs can be adapted to optimize code, achieving performance improvements exceeding those made by human programmers.

In a complementary study, Dou et al. (2023) investigated LLMs' ability to identify duplicate code fragments that can adversely affect software maintainability. They compare LLMs across different clone types, programming languages and

prompting techniques. The authors' findings indicate that LLMs excel in identifying complex semantic clones, surpassing previous methods. These research findings collectively demonstrate that LLMs are becoming an integral part of the software development workflows.

2.3.2 Software quality assurance

The correctness and behavior of software is verified using test cases. There are attempts in software quality assurance to leverage LLMs to generate cases and unit tests (Tang et al., 2024; Wang et al., 2024). These models are capable of generating comprehensive test scenarios. For example, LLMs can be used to generate code that determines whether LLM-generated outputs are semantically correct (K. Zhang, Wang, et al., 2023).

In unit test generation, LLMs are capable of generating test suites for diverse programming tasks (Pan et al., 2024). ChatUnitTest Chen et al. (2024) leverages ChatGPT for automated unit test generation and shows LLMs' ability to create unit tests that validate the expected program behavior.

2.3.3 Assisting in debugging

Debugging is identifying, analyzing, and resolving errors in code. These errors can be syntax errors, logical flaws, or runtime issues. They often require developers to identify relevant components in a codebase, logs, and execution traces. Traditionally, debugging has been time-intensive and demands a deep understanding of the code, expected behavior, and the business context. The use of LLMs has introduced new possibilities for the debugging process. LLMs can interpret logs and map errors to problematic code parts (Tian et al., 2024).

In addition to detection, LLMs can suggest targeted fixes for the identified problems. These range from syntax corrections to logical adjustments that require understanding the code's intent and context (K. Zhang, Li, et al., 2023). Lee et al. (2024) enhance this process by dividing debugging tasks into smaller, manageable sub-problems which enable the LLM to handle more complex debugging scenarios.

Madaan et al. (2023) introduce a method where LLMs generate an initial output, evaluate it to identify potential improvements, and then iteratively refine their responses based on this self-generated feedback. This process resembles iterative improvements performed by humans. Such self-refinement improves the relevance of the output and leads to performance improvements across various tasks.

2.3.4 Requirements elicitation

A requirement is a documented condition, capability, or feature that a system must satisfy to meet user needs, business objectives, or regulatory constraints. It defines expected properties of the environment that the system must bring about or respond to through its interaction with external stakeholders and shared phenomena (“IEEE,” 1990).

Requirements need to be turned into user stories. User stories are often written by product owners, developers, or agile team members. The process is collaborative in nature as capturing end-user needs and prioritizing features requires multiple stakeholders (Marques et al., 2024).

This collaborative requirement elicitation process has traditionally relied on human expertise. However, recent studies have demonstrated that LLMs can support this area despite its cross-functional complexity. Cosler et al. (2023)’s study has

shown that LLMs can assist in converting unstructured requirement descriptions into precise and formal specifications.

A Software Requirements Specification (SRS) is a complete document of specification and description of requirements. SRS typically encompass functional and non-functional requirements. They detail the role of a system and how to achieve them to meet stakeholder needs. Krishna et al. (2024) highlight the ability of the models GPT-4 and CodeLlama to draft SRS documents. The research focused on a university club management system. The LLMs managed to produce structured and sufficiently detailed documents. The findings reveal that the generated documents were comparable in quality to ones crafted by entry-level software engineers. This shows the models' potential for translating user needs into actionable requirements without sacrificing precision.

In software engineering, abstract and informal requirements are the norm, which cannot be formally verified. To this end, Cosler et al. (2023) introduce 'nl2spec' which is an LLM that converts unstructured requirement descriptions into precise, formal specifications. LLMs understand the semantic intent behind the natural language requirements. By parsing and interpreting these inputs, the models generate equivalent formal statements compatible with verification tools, which can then check that the software system meets the unstructured requirement specification.

Li et al. (2024) emphasize employing Chain-of-Thought (CoT) prompting techniques to derive class diagrams from user stories, facilitating the creation of conceptual models in agile requirements engineering. In CoT prompting, the model is guided to generate intermediate reasoning steps before providing a final output.

From drafting SRS and converting unstructured natural language into formal specifications, LLMs have become more visible in the requirement engineering area. Despite these advancements, there are limitations of LLMs in requirement engineering. Contextual understanding of LLMs in addressing ambiguous or multi-layered requirements will be poor and the necessity for human oversight is emphasized (Marques et al., 2024). Additionally, LLMs might produce inconsistent responses to similar prompts, which can compromise the reliability of the requirements elicitation process.

2.3.5 Assisting in documentation

Documentation is part of multiple phases in the SDLC lifecycle. LLMs can help with documentation during the development phase (J. Y. Khan & Uddin, 2023). Code documentation helps developers understand the system behavior and help for future decisions better, which makes the code more maintainable.

LLMs were shown to generate documentation comparable in quality to manually code documentation. Khan and Uddin (2023) investigated the capability of Codex, a GPT-3-based language model trained on large-scale GitHub repositories, to generate documentation automatically. Their study aimed to assess whether Codex could surpass existing models in producing accurate, readable, and informative code comments. Through experiments on six programming languages, Codex performed the best. Their findings suggest that LLMs like Codex can significantly reduce the manual effort required for documentation without compromising quality.

A similar study is conducted by Dvivedi et al. (2024) comparing different language models and their findings suggest that closed-source LLMs, particularly GPT-4 and GPT-3.5, are highly effective for automated code documentation,

offering accuracy and comprehensiveness comparable to or better than human-generated documentation.

Diggs et al. (2024) demonstrated that LLMs were also successfully applied to legacy systems. These models can generate clear, automated code documentation that explains system functionality, making complex legacy code more understandable. This documentation aids in refactoring by helping developers identify outdated patterns and understand system behavior without extensive manual analysis.

2.3.6 Software maintenance

LLMs have been used to refactor code (Almeida et al., 2024; Diggs et al., 2024), find and fix bugs (C. Lee et al., 2024; Tian et al., 2024) as well as to identify technical debt (Maldonado et al., 2017), which refers to the accumulated cost of choosing quick but suboptimal solutions during development that may require additional work in the future.

LLMs can analyze entire code bases and identify areas for optimization. Examples include improving performance or reducing memory footprint (Shypula et al., 2024) . Similarly, LLMs can modernize legacy systems by rewriting outdated code constructs and migrating them to newer frameworks (Almeida et al., 2024; Diggs et al., 2024).

LLMs can also improve code maintainability by simplifying or refactoring code. For instance, Shirafuji et al. (2023) demonstrated the application of GPT-3.5 in proposing simpler versions of Python code that adhere to better coding practices.

Another application for LLMs are automated bug detection and fixing. Hossain et al. (2024) introduced a framework that utilizes LLMs for token-level bug

localization and subsequent repair, effectively addressing issues without prior knowledge of bug locations. The authors were able to identify potential vulnerabilities in code as well as suggesting targeted fixes based on training on diverse programming datasets.

LLMs assist in detecting self-admitted technical debt (SATD) by analyzing code comments and documentation (Maldonado et al., 2017). SATD refers to instances where developers acknowledge suboptimal code implementations or design choices within comments or annotations, indicating areas that may require future improvement (Maldonado et al., 2017). Maldonado et al. (2017) developed a machine learning classifier using natural language processing (NLP) to automatically detect design and requirement-related self-admitted technical debt (SATD) in source code comments. Their approach significantly outperformed traditional keyword-based methods and achieved high accuracy even with a small amount of training data.

While LLMs demonstrate promising capabilities in identifying and managing technical debt, their impact on software maintenance efficiency still requires careful consideration. LLMs can significantly reduce the initial effort in detecting and categorizing maintenance needs, particularly in large codebases where manual review would be time intensive. However, applying improvements to the software systems still requires human expertise to ensure that proposed changes align with the architecture and business requirements.

The preceding sections have detailed various applications of LLMs across different areas of software development. To provide a comprehensive overview of these applications on SDLC, Table 1 synthesizes these applications where they impact the software development lifecycle.

Table 1 LLM applications across the software development life cycle

SDLC Phase	LLM Applications	Citation
Requirements LLMs extract and formalize user needs into structured specifications	Requirement elicitation	Cosler et al. (2023)
	SRS document creation	Krishna et al. (2024)
	User story generation	Marques et al. (2024)
Design LLMs translate requirements into architectural blueprints and design elements	Class diagram generation	Li et al. (2024)
	API design assistance	Belzner et al. (2024)
Implementation LLMs generate, complete, and optimize code based on specifications	Code generation	Chen et al. (2021)
	Code completion	Fan et al. (2023)
	Code optimization	Shypula et al. (2024)
Testing LLMs create test cases and automate verification of software quality	Test case generation	Tang et al. (2024)
	Unit test creation	Chen et al. (2024)
Maintenance LLMs assist in updating, fixing, and improving existing software systems	Bug Fixing	Hossain et al. (2024)
	Code refactoring	Shirafuji et al. (2023)
	Technical debt management	Maldonado et al. (2017)
	Code translation between languages	Roziere et al. (2020)
	Code explanation and comprehension	Diggs et al. (2024)

2.4 Risks of LLMs

The previous sections outlined the practical benefits and considerations of using LLM outputs in different phases of the software development life cycle. This section examines the algorithmic and legal risks of LLMs when used within software development processes. Furthermore, how the awareness of these risks affects LLM adoption is examined.

The focus is narrowed to two aspects of LLM-generated output. To facilitate discussion, an umbrella term 'flawed LLM output' is introduced for LLM output to

cover inaccuracies and legally non-compliant outputs. This classification allows us to systematically examine how flawed outputs will challenge LLM adoption.

2.4.1 Algorithmic risks of LLMs output

2.4.1.1 Accuracy and predictability

LLMs present challenges in software development due to their tendency to generate inaccurate and unpredictable outputs. These issues manifest through incorrect outputs and a phenomenon known as "hallucinations" (Huang et al., 2025), where models produce content that appears valid but contains flaws.

In software development contexts, these inaccuracies appear across multiple domains. For example, (Belzner et al., 2024) discuss the ambiguities and inconsistencies in requirements, incorrect or incomplete test cases generated by LLMs or quality and limitations of generated code and documentation. Liu et al. (2024) provide a framework for understanding these issues by categorizing them into intrinsic hallucinations, where outputs contradict source content, and extrinsic hallucinations, where models generate unverifiable or irrelevant information. These hallucinations can produce code that appears syntactically correct but contains semantic flaws, creating challenges for software development projects (Mohsin et al., 2024).

The implications of these inaccuracies extend beyond surface-level errors. (Mohsin et al., 2024) identified that LLMs can generate code containing security vulnerabilities, creating potential attack vectors for exploitation. This risk is exacerbated by developers' tendency to review AI-generated code less rigorously than human-written code (R. Wang et al., 2024). Furthermore, LLMs can provide misleading explanations that make incorrect answers appear valid (Huang et al.,

2025). When misinformation perpetuates within the development team, this can have ramifications on the project reliability.

The non-deterministic characteristics of LLM outputs present a challenge for code generation. Identical prompts can yield different code outputs across multiple executions. This variability stems from the probabilistic mechanism of these models during text generation (Fan et al., 2023), making it difficult to maintain consistent code generation practices.

Besides code generation, LLMs may produce plausible-looking but technically incorrect documentation, particularly for low-frequency use cases with missing or incorrect parameter information leading to implementation errors (Belzner et al., 2024). This issue highlights the importance of output validation as reliance on hallucinated documentation can disrupt software development processes. Ji et al. (2023) emphasize that hallucination is particularly concerning in sensitive domains like healthcare, where incorrect outputs could have life-threatening consequences.

The challenges of interpretability in AI systems are further exacerbated in that they are often treated as a “black box". These systems being statistical models and having large architectures, complicate the decision-making process’s interpretability. This results in lack of transparency that hinders understanding of their reasoning (Şahin et al., 2025). Ultimately this creates risk of using its output while not knowing the reasoning of LLMs. Ji et al. (2023) highlight LLMs’ hallucinations can produce nonsensical or unfaithful outputs, reducing system performance and failing to meet user expectations in real-world applications such as in medical domains.

This lack of transparency is especially problematic in areas where we need to know why the AI made its decisions. In sensitive domains like healthcare, hallucinated outputs (e.g., incorrect medical summaries or instructions) could lead to serious consequences which can impede adoption due to trustworthiness concerns (Ji et al., 2023).

2.4.1.2 Lack of generalization

LLMs are typically trained for specific tasks and often rely on shortcut learning — strategies that perform well on training data but fail to generalize to novel conditions. Neural networks underperform when presented with out-of-distribution data which occur when shifting between domains in real-world applications. These models are designed under the assumption that training and testing data share the same distribution, which does not necessarily hold in practice (Geirhos et al., 2020).

Although developers tend to use cloud models such as GitHub Copilot, regulatory or organizational constraints may necessitate developers to use local models. These models are often trained on specific programming languages, limiting their ability to perform optimally when applied to different ones (Peng et al., 2024). For instance, a model trained to assist with Python code may struggle to effectively generalize to Java without retraining. This is compounded by the evolution of development practices and new technologies, which can render AI models less effective over time. Peng et al. (2024) highlight the difficulties in generalizing across different programming languages, emphasizing the need for retraining to achieve proficiency in multiple languages.

2.4.1.3 Effect of data flaws in LLMs

The effectiveness of LLMs is inherently linked to their training data quality. Most SDLC phases utilize cloud APIs, with model training and retraining typically depending on third parties. In this common arrangement, organizations have limited or no control over the training data quality used to develop these models. As the accuracy of LLM outputs depends heavily on the comprehensiveness of their training datasets (Vig et al., 2020), when these datasets contain flaws, biases or gaps, the resulting models reflect these limitations in their outputs (Vig et al., 2020). Subpar data quality can lead to harmful outcomes in software systems such as incorrect medical advice due to biased or incomplete training data (Ji et al., 2023). Garcia et al. (2024) documented an example in mortgage lending where AI systems demonstrated racial biases in loan application processing, resulting in disproportionate denial rates for minority applicants compared to equally qualified white applicants due to the bias in the training data set.

The accessibility of externally managed LLMs through third-party APIs introduces additional risks. API providers may operate with outdated knowledge bases, potentially affecting the reliability of AI outputs as the recency is much lower. This time lag between model updates means users receive outputs based on outdated information. The usefulness of AI-driven outputs thus depends not only on the model's capabilities but also on the update frequency and maintenance of these API services (Wu et al., 2024).

However, certain sectors face unique challenges that necessitate a different approach. In scenarios where proprietary data is available or where cloud APIs cannot be used due to privacy considerations, organizations opt for on-premise LLMs (Wiest et al., 2024). While this provides greater control over data quality, it

introduces its own set of challenges such as model maintenance and retraining requirements ultimately causing high computational costs. Wu et al. (2024) highlight that retraining LLMs incurs significant financial costs while being necessary for maintaining their long-term effectiveness. As software development evolves rapidly, frequent model updates are needed to ensure accurate code generation that aligns with current practices. They emphasize that LLMs require regular fine-tuning to keep pace with evolving programming languages, libraries, and frameworks.

2.4.1.4 Limited context awareness

LLMs often operate within the context of their training data and fail to fully understand nuanced requirements in real-world scenarios and may struggle to document stakeholder needs. This limitation is evident in domain-specific applications where LLMs lack context to generate accurate output. Ji et al. (2023) highlight that LLMs can hallucinate due to parametric knowledge bias which causes models to rely on memorized patterns rather than grounding outputs in the input context. Particularly in specialized domains due to insufficient contextual understanding. This can lead to incomplete or ambiguous requirements which will affect subsequent development stages. For instance, while LLMs can assist in generating Software Requirements Specifications, their outputs may lack the necessary depth and precision and require extensive human review (Krishna et al., 2024).

2.4.2 Legal risk of LLM outputs

Integrating LLMs in software development introduces legal challenges. As AI systems are increasingly sophisticated and widely adopted, legal implications

become a concern for software developers and organizations. As LLMs are trained on vast datasets and generate content based on learned patterns, legal uncertainties arise. In software development, code generation must comply with legal frameworks that were designed before the advent of AI.

This section examines the legal risks associated with LLMs in software development and explores various legal dimensions and their implications for practical implementation.

2.4.2.1 Copyright and intellectual property concerns

Copyright and intellectual property concerns are the main legal challenges in the context of LLM outputs. The issue stems from how LLMs generate content by learning from vast training datasets, including copyrighted materials. Therefore, AI may unintentionally replicate copyrighted content in its outputs. Liu et al. (2024) demonstrated that current LLMs can output copyrighted text which has led to several high-profile lawsuits (Alter & Harris, 2023; Grynbaum & Mac, 2023; Vallance, 2023).

Instances where LLMs have reproduced copyrighted code, as documented by Ma et al. (2024) and Xu et al. (2024) raise questions about the legality of their outputs. (Ma et al., 2024)'s study reveals that AI powdered programming language generation (PLG) models generate code that is identical to licensed source code without proper attribution. The paper explains that these models are essentially memorizing code snippets from their training data, which includes licensed source code from repositories like GitHub. For instance, GitHub Copilot has been reported to produce substantial portions of code directly copied from licensed sources without proper attribution, resulting in a copyright infringement lawsuit. The complexity is

compounded by the fact that copyright law was not designed to address AI-generated content.

Similarly, Karamolegkou et al. (2023) found that large language models memorize copyrighted text fragments, as well as complete coding problems (e.g., LeetCode descriptions). This memorization can occur at various levels, from direct code reproduction to structural similarities that might still infringe on copyright protection.

In terms of SDLC, there are discussions about the legal ownership of AI-generated code and accountability for AI-related errors. Traditional copyright laws typically grant authorship to human creators, leaving AI-generated works in a legal gray area. Matulionyte and Lee (2022) discuss the possibility of attributing copyright of AI output to the owner of the AI system, allowing the individual or entity controlling the AI to claim ownership over its outputs. This aligns with the notion that the level of control over the model and the intent in using the system are of importance in ownership claims. While some countries have begun to work on addressing AI-generated content in their copyright frameworks (Matulionyte & Lee, 2022b), the global landscape remains unclear. This uncertainty extends to questions of authorship, duration of protection and the requirements for copyright protection of AI-generated code.

On the other hand, the deployment of AI systems introduces challenges in assigning accountability for errors. Novelli et al. (2024) relate accountability in AI to answerability. Answerability is the requirement for an entity to justify their actions and decisions generated by AI systems. The necessary conditions of answerability are authority recognition, interrogation, and power limitation. Authority recognition means that the entity responsible for the AI system must be known and, in a position,

to answer questions. Interrogation mechanisms must be in place to investigate AI-related decisions or errors with responsible parties. Power limitation ensures that no entity exercises unchecked authority to prevent power abuse.

Copyright, ownership, and accountability are legal considerations that software developers and stakeholders must address when incorporating LLM outputs into their projects. With the legal framework and technology evolving, organizations are required to keep adjusting their approach to mitigating legal risks. The following section will examine privacy-related risks associated with LLM implementation.

2.4.2.2 Data privacy, confidentiality, and security issues

Data privacy, confidentiality, and security concerns in LLM outputs extend beyond simple data protection and encompass regulatory compliance issues and business confidentiality requirements. Recent work has demonstrated that LLMs can inadvertently reproduce personal information (Borji, 2023; Carlini et al., 2021) or sensitive business data (Carlini et al., 2021; W. Z. Khan et al., 2024) included in their training data or which creates significant privacy risks. (Peng et al., 2023) highlight that ChatGPT has been banned after a sensitive code leak from a private company. When an LLM outputs contain personal data from the training set, this violates privacy laws, such as the "right to be forgotten" and data protection regulations like GDPR (General Data Protection Regulation (GDPR) – Legal Text, n.d.) or CCPA (California Consumer Privacy Act of 2018, n.d.) (Borkar, 2023; Illman & Temple, 2019).

Security risks could affect generated code. Chen et al. (2021) highlighted instances where LLM-generated code contained security vulnerabilities. (Borji, 2023) highlights that this makes the model a potential target for attacks that can

affect any applications derived from it. Similarly, models were found to generate SQL queries that were vulnerable to SQL injection attacks, and web components with Cross-Site Scripting (XSS) vulnerabilities (Varghese & Thomas, 2024). Another security vulnerability was found when models suggested an authentication mechanism that could be easily bypassed (Varghese & Thomas, 2024).

Cloud LLMs are trained on user prompts to better perform in the future. The UK National Cyber Security Center (ChatGPT and Large Language Models, n.d.) highlights that queries are visible to organizations providing the LLM, and those queries will be used as training data for continued model improvements. (Zhao et al., 2024) publish the WildChat dataset which comprises over 1 million real-world user-chatbot interactions with ChatGPT and GPT-4 which were collected with user consent. Even though personally identifiable information was removed or masked for user anonymity, the authors highlight that there is still a risk of users sharing sensitive data as it may not be covered by their anonymization efforts.

When LLMs are trained on user prompts, these models may reproduce confidential information such as personally identifiable information, project, or organization secrets. This creates liability not only for the organizations whose data might be exposed but also for developers who incorporate LLM outputs into their projects. Ma et al. (2024) emphasizes the importance for developers to be aware of data protection laws and copyright issues when incorporating AI-generated code into their projects.

LLMs often require large datasets. This creates challenges for compliance with regulations like GDPR and CCPA. These regulations have strict requirements on how personal data is collected and stored. Ultimately creating hurdles for organizations attempting to gather data for training purposes. As Şahin et al. (2025)

mentioned, AI challenges GDPR's principle of data retention limits. Deleting data after its original purpose conflicts with AI's need for long-term use for future training passes. They further explore the explanatory difficulties arising from AI's 'black box' nature, which complicates efforts to provide clear information about data processing methods and decision-making factors. Their paper highlighting the difficulty of complying with GDPR's transparency mandates and AI's inherent opacity.

Consequently, balancing the need for large datasets while adhering to privacy laws is a challenge when training AI models. Finding equilibrium between comprehensive dataset requirements and privacy regulation adherence remains a critical issue in AI model development. Non-compliance can have legal ramifications. The following cases have shown that organizations can face not only financial penalties but also lasting reputational damage when failing to properly address legal concerns (See Table 2).

This emphasizes the balancing act between restrictive individual privacy rights while still allowing technical progress and AI innovations to occur. It can be expected that policymakers will continue to adjust laws accordingly.

Table 2 summarizes some regulatory actions and fines imposed on AI companies over privacy, data protection, ethical and transparency violations across different jurisdictions.

Table 2 AI Companies Fined for Regulatory Non-Compliances

Jurisdiction	Case	Penalty	AI Risk	Resource
Italy	Clearview AI fined for biometric data scraping	€20,000,000	Privacy/Transparency	(Hern, 2022)
United States	Lawyers fined for citing falsely generated ChatGPT cases	€4,600	Transparency/efficacy	(Milmo, 2023)
Italy	Italy's privacy watchdog fines OpenAI for ChatGPT's violations in collecting users personal data	$15.6 million	Privacy	(Zampano, 2024)
United States	'Robot lawyer' company faces $193,000 fine as part of FTC's AI crackdown	$193,000	Legal/Ethical	(Vasani, 2024)

2.4.2.3 Bias and discrimination

AI models trained on biased data can lead to discriminatory outcomes (Vig et al., 2020). Organizations may face legal action when these biases result in unfair treatment of users. For instance, Amazon discontinued its AI recruiting tool after discovering it favored male candidates due to biases present in the training data (Dastin, 2022).

Fuster et al. (2022) analysed how advancements in machine learning have introduced biases in credit markets. While majority groups (e.g., White non-Hispanic borrowers) often benefit from improved predictive accuracy, minority groups (e.g., Black and Hispanic borrowers) did not. The models amplify existing disparities in the underlying data due to their "flexibility" and "triangulation" properties. In Fuster et al. (2022)'s study, flexibility explained as advanced statistical models can use permissible variables (like income or credit score) to predict outcomes (like default risk) with high accuracy. However, this will disadvantage individuals that have less favorable values for these variables due to existing group disparities. Triangulation

refers to the model's ability to infer hidden variables like race or gender from unrelated data. Thus, both data biases and the model design can cause discrimination by amplifying existing inequalities present in the data.

Similarly, Dressel and Farid (2018) explore risk assessment algorithms used in the U.S. criminal justice system. They discuss how predictive models like COMPAS can disproportionately affect marginalized groups, causing Black defendants to be twice as likely falsely classified as high risk compared to white defendants. Similarly, white defendants were more often mislabeled as low risk.

Borji (2023) asked ChatGPT whether it is aware of its own biases. The response they received declined this: "As a language model, I have been trained on a large dataset of text that includes many examples of biases and stereotypes. While I have been designed to recognize patterns in this data and generate text based on these patterns, I do not have the ability to recognize or understand my own biases. I do not have consciousness, self-awareness, or the ability to reflect on my own beliefs and values. Therefore, it is important to be mindful of potential biases in the text I generate and to take steps to counteract these biases, such as by including a diverse range of perspectives and sources of information." (Borji, 2023, p.6)

The production of potentially biased outputs by artificial intelligence can erode confidence among both regulators and end-users towards AI systems. Consequently, decision-making clarity becomes essential for trust preservation. Fisher et al. (2013) noted that transparency serves dual purposes—building user confidence while simultaneously enabling proper validation of system-generated content.

To tackle these issues, experts across academia and industry have suggested several risk management approaches for AI deployments. The next section explores

these mitigation strategies, with particular attention to how they fit within development lifecycles.

2.5 Mitigation strategies

The case studies mentioned in previous sections largely point to the numerous practical benefits of using AI assistants in SDLC processes. They can boost an organization's competitiveness by boosting productivity and quality. However, the introduction of LLMs also bears risks. Mitigation strategies have emerged, ranging from developer training (Varghese & Thomas, 2024a) to improved data management practices (Borkar, 2023). In particular, rigorous review processes by human developers can mitigate legal and security risks (Wang & Chen, 2023). Data governance practices implemented at the organization level are relevant for privacy risks. This includes the management of training data and regular audits of AI system outputs. Borkar (2023) suggests companies remove data points at a higher risk of containing confidential information to comply with the 'right to be forgotten' policy within GDPR.

Wang and Chen (2023) bring attention to the scarcity of research on human engagement when evaluating AI-generated code. As the field evolves, the balance between AI's capabilities and the associated risks remains a consideration for software development teams. An approach that combines technical expertise and legal compliance to responsible AI use can determine success. Moving forward, human involvement when evaluating AI output will be explored in more detail.

2.5.1 Human oversight

Over-reliance on AI in software development may lead to reduced human oversight (Kuhail et al., 2024) . Pearce et al. (2022) found that approximately 40% of code suggestions by GitHub Copilot contain security vulnerabilities which is a significant risk factor when relying on AI without adequate oversight. A similar result from a more recent study by Fu et al. (2025), also identified roughly 29.5% of Python snippets and 24.2% of JavaScript snippets produced by AI assistants had security issues. This concerning statistic highlights the need for proper validation of AI-generated code in software systems. Otherwise, insufficient review processes can introduce critical vulnerabilities that might remain undetected.

The European Union's AI Act defines human oversight as measures enabling people to fully understand the capabilities and limitations of AI systems. Human oversight requires the ability to monitor operation, detect anomalies and intervene or stop the system when necessary. When using AI technologies, human intervention ensures that they operate in alignment with legal requirements as well as security needs. The EU’s AI Act demands human oversight for high-risk AI applications (Article 14, n.d.-a). It mandates the ability for humans to intervene.

Methnani et al. (2021) developed a novel framework called variable autonomy (VA) which dynamically adjusts AI autonomy levels that can range from full autonomy to complete human control. Their approach helps creating a collaborative human-AI system where appropriate levels of oversight can be maintained while maximizing the benefits of automation.

Oversight may be achieved through governance mechanisms such as human-in-the-loop (HITL), human-on-the-loop (HOTL), human-in-command (HIC) or human-out-of-the-loop (HOOL). The European Commission (2019) suggested organizations to

implement HITL, HOTL and HIC approaches as effective governance strategies to maintain appropriate human oversight on AI systems. HITL refers to the capability for direct human intervention during an AI system's operation, allowing humans to validate decisions or provide input before the system proceeds with a task (Nahavandi, 2017). However, this is not always feasible in practice as HITL can significantly slow down development processes and increase costs, especially in scenarios requiring rapid iterations or real-time responses. HOTL, on the other hand, focuses on enabling humans in a monitoring or supervisory role during the system's operations and intervening if failures occur (Nahavandi, 2017). HIC refers to the capability to oversee an AI system's overall activity while considering its societal, economic, legal, and ethical impacts. This approach empowers humans to define and manage the operational boundaries of AI systems (Aschenbrenner & Colloseus, 2023). HOOTL removes humans from direct engagement in system control loops (Kaber & Endsley, 2004). For example, advanced driver-assistance systems can identify potential collision risks, alert the driver, and take control of the car if necessary to prevent accidents. They can enhance road safety and the need for full autonomy of these systems arises from the limited human reaction time, possible human errors, or lack of situational awareness. By removing the human entirely from the control loop, the system can carry out tasks independently (Methnani et al., 2021). These models allow organizations to control the level of human oversight for different phases of the SDLC process.

According to Langer et al. (2024), effective oversight is a balance between detecting legitimate issues while minimizing false alarms, which is achieved through monitoring processes and explainable AI (XAI) methods. Organizations can solve this with training programs and enhance oversight personnel's ability to recognize

and address potential ethical violations. Through human oversight, organizations can implement accountability measures that ensure AI systems decisions align with ethical requirements and do not perpetuate discriminatory practices against any groups.

Human oversight is a factor in mitigating AI risks. Its success also depends on organizational as well as technical factors. Practical frameworks that incorporate rigorous human oversight involvement in AI processes, can reduce the risk for legal disputes a company faces.

2.5.2 Legal frameworks

The growing prevalence of LLMs requires robust legal structures for risk mitigation. With AI technologies becoming more complex and embedded throughout software development lifecycles, proper regulatory supervision has become more important. Legal frameworks protect not only stakeholders but also provide guidelines for organizations implementing LLM technologies.

Legal frameworks can be challenging to develop as the decision making of LLM models is non-deterministic, models are not easily interpretable, and they have a wide range of applications. Furthermore, systems generate content with legal issues from different jurisdictions and legal domains, such as intellectual property to data privacy. Furthermore, AI requires that regulatory frameworks can provide meaningful protection and are flexible enough to accommodate rapid technological advancement.

2.5.2.1 Current regulatory landscape

There are different approaches to AI regulation in different countries. Several major initiatives have emerged in recent years. For instance, the European Union's AI Act, introduced in 2021, represents the first comprehensive attempt to regulate AI systems (Li, 2023). This AI Act is planned to be enforced progressively over the coming years. The legislation adopts a risk-based approach, categorizing AI systems into four levels: minimal risk, limited risk, high risk, and unacceptable risk (Artificial Intelligence Act, n.d.).

The AI Act put in place the concept of 'provider accountability' holding individuals or entities involved in the development, deployment, or operation of AI systems responsible for their actions. In doing so, it establishes stringent obligations for both AI developers and deployers, while also seeking to ease regulatory burdens on businesses.

By implementing the AI Act, the EU aims to establish a balanced regulatory environment. It is important to keep the innovative environment fostered, at the same time safeguarding public interests (Nicholas & Friedl, 2024). The European Union's AI Act represents a pioneering framework, but there are also several other regulatory initiatives that have emerged worldwide.

The United States has a more decentralized approach including federal executive orders, agency guidelines, and state-level legislations. (Hacker et al., 2025). They have introduced the Executive Order on Safe, Secure, and Trustworthy AI guidelines in 2023 which was revoked in early 2025. The new guideline is Executive Order on Removing Barriers to American Leadership in Artificial Intelligence, focused on maintaining U.S. global AI leadership (Hacker et al., 2025). There is an absence of binding federal AI laws across states. Thus, individual states

have enacted their own regulations. Besides, there are agencies with focus on introducing frameworks to regulate AI such as the National Institute of Standards and Technology (NIST). They have issued voluntary frameworks to guide responsible AI development. NIST's AI Risk Management Framework (AI RMF - AIRC, n.d.) aims to provides organizations principles and practices for managing AI-related risks. There are also other initiatives at local governments level such as New York City's Local Law 144, effective from July 2023 targeting specific AI risks. This law mandates that automated employment decision tools undergo independent audits for bias before they can be used in hiring or promotion processes (Hacker et al., 2025).

There are other regulations from different countries. China has enacted the Interim Measures for the Management of Generative Artificial Intelligence Services which were enacted in 2023. This regulation mandates that training datasets must adhere to intellectual property rights and respect individual privacy protections. Furthermore, the measures requiring AI-generated content must be clearly labelled to distinguish it from human-created content. The regulation also forces companies to register their AI algorithms with regulatory authorities for transparency and accountability. Under the Interim Measures, providers are held accountable for the content generated by their AI systems (Migliorini, 2024). China set a goal to lead in AI development by 2030. Therefore, they have a focus on creating robust ethical standards and participating in global regulatory frameworks (Migliorini, 2024).

While the regulatory examples presented in this section provide insights into the global trajectory of AI governance, it should be noted that this review is not exhaustive. Only a selected number of representative regulations from different regions were included.

2.5.2.2 Gaps and limitations

Despite these regulatory initiatives, there are gaps in the legal framework surrounding LLM usage as laws are largely in their infancy and may not fully address the challenges posed by LLM implementation in business processes. A "pacing gap" exists between the rapid change of technology and the slower regulatory responses, exemplified by the Collingridge dilemma—limited data at technology's emergence and outdated regulations by the time data is available (Goanta et al., 2023). McIntosh et al. (2024) evaluate four cybersecurity frameworks for their readiness to address opportunities, risks, and regulatory compliance in adopting LLMs. The study findings indicate that all frameworks lack comprehensive strategies for LLM-specific risks. Most frameworks inadequately address the challenges of AI system misclassification, hallucinations, and bias. Li (2023) identifies several critical limitations in the current European AI Act (AIA) as AIA's transparency requirements, such as informing users that they are interacting with AI are insufficient to address hallucination risks. Li (2023) additionally analysis the Digital Services Act (DSA) and Digital Markets Act (DMA) highlighting additional regulatory gaps. The DSA's primary focus on intermediaries overlooks the crucial roles of LLM developers and their outputs, while the DMA's emphasis on market structure and gatekeepers results in insufficient attention to AI-specific risks, creating a significant gap in comprehensive AI governance.

2.5.2.3 Recommendations for future framework development

Developing viable legal safeguards against LLM risks demands multi-dimensional coordination, such as technical, ethical, industry, and so on. Recent studies offer

thorough guidance on strengthening LLM governance structures, stressing the importance of balancing technical considerations with human-centered design principles. Li (2023) advocates for fundamental revisions to the AIA's risk taxonomy, suggesting a shift beyond basic transparency requirements to incorporate trustworthiness, reliability, and explainability as core regulatory benchmarks. This recommendation aligns with McIntosh et al.'s (2024) emphasis on enhanced transparency in LLM outputs and data sources, along with their call for continuous bias testing and model validation mechanisms that align with the EU AI Act provisions for real-time safeguards and automated compliance checks. To address market-related concerns, Nicholas and Friedl (2024) propose the development of watermarking mechanisms for detecting AI-generated content, which could help identify misinformation. They also highlight the risks of market concentration, such as monopolies across the LLM supply chain. They warn regulators about reduced competition and data exploitation by dominant firms. Li (2023) further emphasizes the importance of user protection, recommending the establishment of avenues for contesting inaccurate AI outputs and imposing user responsibilities for content verification in sensitive domains such as healthcare and law. For developer accountability, Li (2023) suggests mandatory implementation of safeguards including rigorous testing, real-time dataset updates, and explicit warning systems about potential inaccuracies. Their study also recommends that LLMs should be programmed to redirect users to authoritative, traceable sources rather than generating potentially unreliable responses.

The recommendations outlined above provide a foundation for building such comprehensive regulatory structures, though they will need continuous refinement as our understanding of LLM capabilities and risks continues to evolve.

CHAPTER 3

MODEL DEVELOPMENT AND HYPOTHESIS

3.1 Adoption theories

To understand the factors that influence LLM adoption in software development teams, I will examine established adoption theories, in particular the Protection Motivation Theory (PMT) by Rogers (1975) and the User Acceptance of Information Technology Model (UTAUT) by Venkatesh et al. (2003).

3.1.1 Protection motivation theory

PMT explains how individuals respond to persuasive messages involving threats or risks. Proposed by Rogers (1975), PMT has been widely adopted in behavioral adoption studies because it represents a significant advancement in understanding human responses to threats. Before PMT's development, research primarily examined fear appeals, which is a persuasive communication strategy designed to trigger anxiety through emphasizing potential dangers to modify behavior. Such fear-appeal strategies typically identify particular risks, stress the target audience's vulnerability to these risks, and often suggest protective measures to reduce the threat (Ruiter et al., 2014). PMT shifted focus from fear alone to a more complex understanding of human decision-making, not only focusing on threat appraisal but also including coping strategies in its model. It emerged from combining the insights of expectancy-value theories and cognitive processing theories to explain how fear appeals influence behavior (Ifinedo, 2012). PMT posits that the intention to adopt a recommended protective behavior is a function of two cognitive processes: threat appraisal and coping appraisal. Threat appraisal process involves two constructs:

First construct is the perceived severity of the threatened event. Perceived severity refers to an individual's subjective assessment of the seriousness of the negative consequences associated with the threat. Second construct is the perceived vulnerability to the event's occurrence. This refers to an individual's subjective assessment of the probability that they will experience the threatened event. On the other hand, the coping appraisal process focuses on the assessment of the recommended coping response and consists of two constructs: First construct is the perceived response efficacy which is the belief that the recommended coping response will effectively prevent the threat. Second construct is the response cost, refer to any costs or barriers related with carrying out the recommended coping response. These costs are incurred from taking the protective action and can range from financial expenses, time, effort, discomfort, inconvenience, to negative side effects.

PMT assumes that these cognitive processes combine to form protection motivation which is the intention to perform the recommended protective behavior. The theory originally proposed a multiplicative relationship between the threat appraisal and coping appraisal constructs in determining protection motivation. Multiplicative interaction assumes that protection motivation is strongest when both threat appraisal and coping appraisal are high.

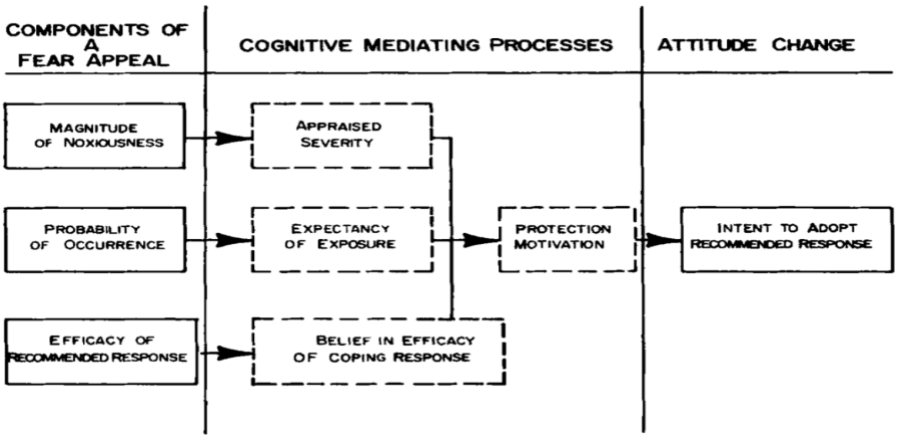

Figure 2 Schema of the protection motivation theory (Source: Rogers (1975))

In a revised version of PMT, Rogers and Maddux (1983) made several changes to the original theory. First, the revised PMT expanded the sources of information that can initiate fear appeals, such as prior experience, observational learning, and personality variables. Second, the revised theory drew a clearer line between two types of coping: adaptive and maladaptive coping responses. Adaptive coping strategies referring to actions aimed at preventing the threat and maladaptive coping referring to easing emotional distress caused by the threat without addressing the threat itself.

Third, the revised PMT separated self-efficacy from response efficacy as distinct concepts. The updated model positioned self-efficacy as one of five cognitive factors—alongside perceived severity, perceived vulnerability, response efficacy, and response cost—that shape decision-making. This addition reflected (Bandura, 1978) influential work highlighting self-efficacy's critical role in behavior change.

The final significant shift abandoned the original theory's multiplicative approach. Instead, Rogers and Maddux (1983) proposed that threat evaluation and coping assessment constructs combine additively or in various combinations to determine one's motivation for protective action. The components do not need to

amplify each other as in a multiplicative interaction but instead have independent, cumulative impacts on protection motivation.

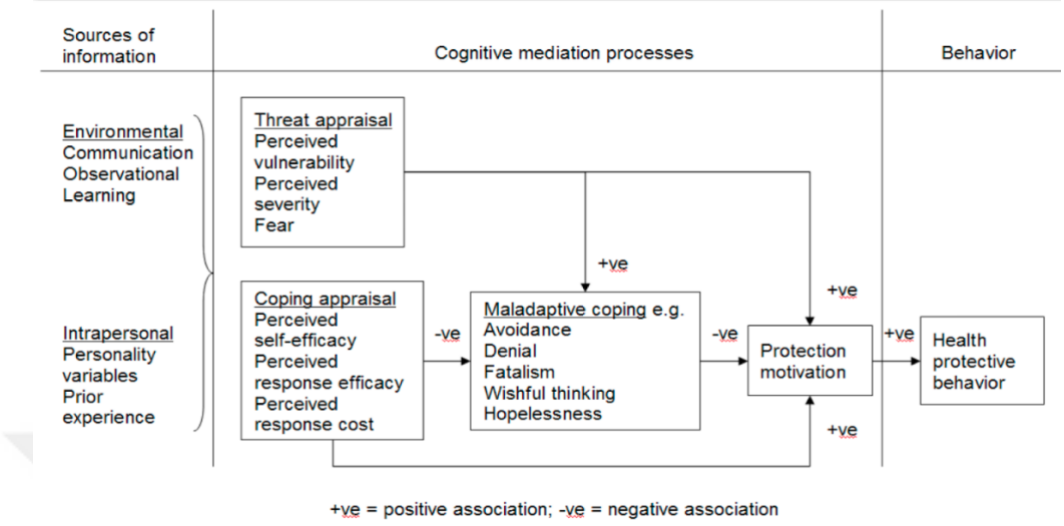

Figure 3 Adapted pmt (Source: Milne et al. (2000))

In summary, PMT reveals that people facing threats employ a dual assessment approach. They begin by assessing the threat itself—weighing its potential severity, their personal exposure, and chances of encountering it. Following this initial appraisal, they scrutinize possible responses by considering both whether the suggested coping response actually work and their personal ability to carry out such protective actions.

Having originated in health psychology, PMT have since been applied to technology adoption contexts due to fundamental parallels between health threats and technological risks (Liang & Xue, 2009). These parallels manifest in three ways, as (Chenoweth et al., 2019) summarize this as: First, both contexts deal with external factors that can have detrimental effects whether on human health or technological systems. Second, the impact and perception of these threats vary among individuals, with different people experiencing and interpreting the same threat in distinct ways.

Third, both domains offer various coping mechanisms to address these threats. This similarity in threat response led (Liang & Xue, 2009) to assert that "People experience similar cognitive processes in response to either health threats or IT security threats."

PMT stands as a most effective theory in understanding behavioral intentions with regards to protective measures (Anderson & Agarwal, 2010). Although PMT first emerged in the medical domain, it was widely adopted in the information security (ISec) field (Crossler & Bélanger, 2014). Boss et al. (2015) highlight that PMT is naturally suited for ISec contexts as users require additional motivation to protect their information assets. Building on this foundation, (Johnston & Warkentin, 2010) further adapted PMT principles to information security through their development of the Fear Appeals Model (FAM). Their research demonstrated that fear appeals significantly impact users' behavioral intentions to comply with security measures, though this impact varies based on individual perceptions of threat severity, self-efficacy, response efficacy, and social influence.

In the context of personal security, (Zhang & McDowell, 2009) applied PMT to understand the drivers for password protection behaviors. Their findings revealed that response efficacy and fear were the strongest predictors for using strong passwords, while response costs had an adverse effect. Their study found that perceived severity and vulnerability, which are central to PMT, did not significantly influence password protection intentions. (Ifinedo, 2012) studied PMT in organizational settings to examine employees' compliance with information security policies. The theory has helped explain why some employees adhere to security guidelines while others engage in risky behaviors.

PMT has proven effective in information security. Recent research has extended its application to artificial intelligence adoption in organizational settings (Chen et al., 2024). The authors reframe AI implementation as a potential threat that triggers protection motivation mechanisms, specifically through what they term "Promethean anxiety", referring to employees' perceived need to safeguard the hierarchy between humans and machines (Bajohr, 2022). Chen et al. (2024)'s work uses PMT's cognitive appraisal framework to explain how AI adoption triggers employees' need for occupational self-actualization as a coping response and subsequent intention to resign from their role. This application of PMT to multiple fields demonstrates its explanatory power in relation to human responses to technological changes.

The emerging research on PMT applied to AI has already recognized the need for understanding protective behaviors in response to AI implementation (Chen et al., 2024). While AI technologies have seen rapid and widespread adoption across various domains, researchers have begun to identify the risks and negative implications of AI usage (Borji, 2023). However, user's perception and response behavior towards these AI-related risks remains limited, particularly in the context of software development.

Prior studies on PMT focus primarily on security behaviors, while AI adoption studies focus on the technical aspects of AI. This leaves a gap for researching users' threat perceptions and protective responses towards AI risks. This study addresses this gap by examining how users perceive and respond to threats associated with AI usage. PMT is utilized as a theoretical framework for understanding their protective behaviors. Specifically, how users' threat appraisals

and coping mechanisms influence their intention to adopt AI, more specifically LLM, technologies are investigated.

Marikyan and Papagiannidis (2023) highlights that different studies have produced varying results when applying Protection Motivation Theory (PMT) in different contexts. For instance, when investigating users' intentions to back up their data, only self-efficacy and response efficacy were found to have a direct influence (Crossler, 2010). In the study of secure online behavior, neither perceived threat severity nor self-efficacy showed significant correlation with behavioral intention (Tsai et al., 2016). Similarly, when examining the adoption of anti-spyware software, all PMT constructs except self-efficacy were found to be significant predictors of behavioral intention (Chenoweth et al., 2019).

These inconsistencies can be attributed to the differences in how individuals perceive threats and their potential consequences, which vary depending on the specific context in which the behavior is studied (Ifinedo, 2012; Verkijika, 2018). Therefore, it is essential to extend PMT by incorporating context-specific factors to enhance its predictive power (Ifinedo, 2012; Thompson et al., 2017; Verkijika, 2018).

Therefore, as PMT is mainly focusing on the risk perspective of adoption intention, in this study benefits of AI usage included as an influencing factor for adoption intention. The benefit constructs are taken from the UTAUT model by (Venkatesh et al., 2003).

3.1.2 Unified Theory of Acceptance and Use of Technology (UTAUT)

The Unified Theory of Acceptance and Use of Technology (UTAUT) represents a milestone in technology acceptance research developed by (Venkatesh et al., 2003).

It stands as one of the most extensively used theories that have been validated repeatedly across different context and technologies. UTAUT synthesizes eight technology acceptance models into a unified theoretical framework namely: Theory of Reasoned Action (TRA), Technology Acceptance Model (TAM), Motivational Model (MM), Theory of Planned Behavior (TPB), Combined TAM and TPB (C-TAM-TPB), Model of PC Utilization (MPCU), Innovation Diffusion Theory (IDT), Social Cognitive Theory (SCT).

Venkatesh et al. (2003) identified four principal drivers of users' acceptance and technology engagement, along with four moderating factors. Core constructs of UTAUT are performance expectancy, effort expectancy, social influence, facilitating conditions and the moderating factors are gender, age, experience, voluntariness of use.

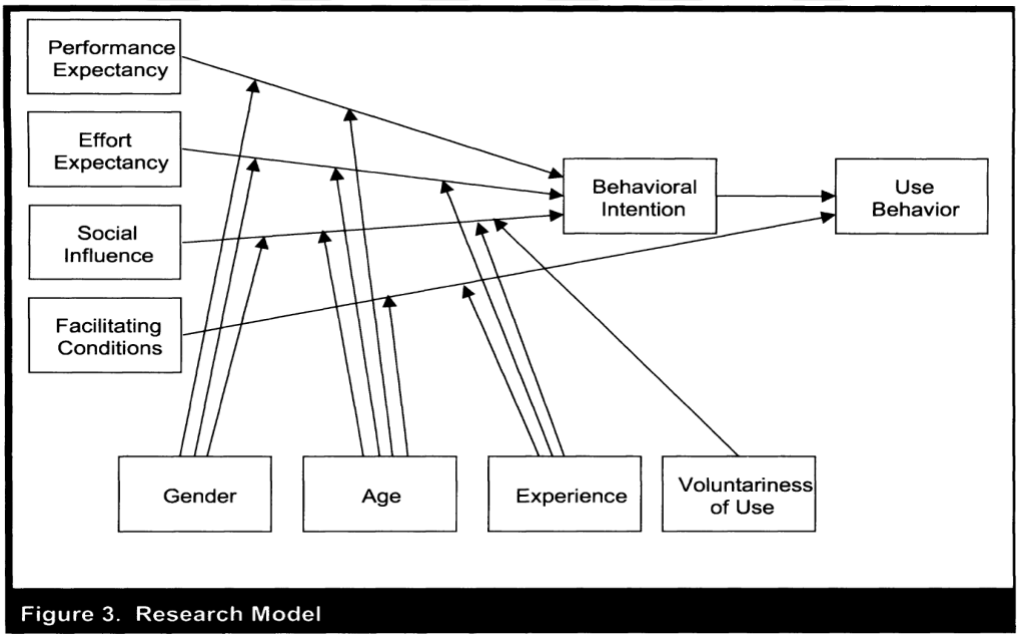

Figure 4 Utaut research model (Source: Venkatesh et al. (2003))

The fundamental constructs of UTAUT offer explanatory power for technology adoption decisions such as performance expectancy, which captures

users' beliefs about productivity gains, has consistently emerged as the most powerful predictor of adoption intention across studies (Dwivedi et al., 2019). Effort expectancy is the perceived complexity of new technology use where adoption decisions impact workflow efficiency. Social influence reflects organizational dynamics and peer effects, while facilitating conditions encompass both technical infrastructure and organizational support necessary for successful adoption.

Building on the first model, Venkatesh et al. (2012) extended UTAUT to explore consumer acceptance contexts, adding three factors: hedonic motivation, price value, and habit.

Recent literature reviews show that UTAUT has evolved to include additional variables such as self-efficacy, perceived trust, perceived risk, and anxiety (Williams et al., 2015) with variations depending on the technology, industry, and combined model. Further studies incorporated risk-related constructs and trust factors extensively (Slade et al., 2015; Tamilmani et al., 2021). While Tamilmani et al. (2021) documented UTAUT2 effectiveness across various organizational settings, subsequent research explored its application in AI contexts. Vimalkumar et al. (2021) examined its utility in understanding AI technology adoption despite the privacy concerns, while (Cao et al., 2021) investigated managerial perceptions towards using AI for decision-making processes, identifying specific risks and concerns affecting adoption intention. Building on the scholars, Venkatesh (2022) demonstrated that while traditional UTAUT constructs remain relevant for AI adoption, they must be contextualized within the technology's, individuals' and environments' unique characteristics. Another recent meta-analysis about UTAUT (Blut et al., 2022) has highlighted this very point, indicating that UTAUT's core constructs may not be sufficient to explain AI adoption suggesting to incorporating

trust in AI systems, algorithmic transparency, ethical concerns and fairness, regulatory compliance and perceived risk into UTAUT when analyzing AI adoption. The study argues that UTAUT research has reached a saturation point, with many studies making only incremental contributions instead of offering new theoretical advancements.

The authors suggest moving towards a "blue ocean" research approach by developing new theoretical frameworks beyond UTAUT for AI-related studies. In addition, Alneyadi et al. (2023) calls for combining UTAUT and PMT to better explain users' intention to adopt AI-based systems. This study answered their call for combining these two theories in order to explain user intention to adopt LLMs in software development teams.

In short, although UTAUT has been effective to understand technology adoption behaviors primarily focusing on the benefits and facilitating conditions that drive acceptance, the adoption of emerging AI technologies like LLMs presents challenges related to output reliability and legal concerns that UTAUT alone may not fully capture. Therefore, UTAUT with PMT is combined to address this limitation by incorporating both risk appraisal and coping appraisal mechanisms into the theoretical framework. Recent studies have demonstrated the value of this integrated approach (Al-Sharafi et al., 2023; Fu et al., 2024; Hsu & Silalahi, 2024; Ong et al., 2024).

Fu et al. (2024) combined UTAUT and PMT to examine ChatGPT adoption in higher education. Their study argues that while UTAUT can capture technology acceptance factors like performance expectancy and effort expectancy, AI technologies necessitate consideration of AI risk factors in the educational context. The authors show that adoption is influenced by perceived benefits such as task

efficiency and hedonic motivation work in conjunction with threat appraisal and coping mechanisms. Fu et al. (2024) focus on the key threat appraisal as risk of being exposed to misused ChatGPT during studies and impact on academic performance and reputation if ChatGPT is misused. They defined the coping response as the participant's belief in their capability of using ChatGPT for academic tasks and confidence in their ability to use ChatGPT appropriately.

Another domain affected by LLM adoption is software development. Unlike education where concerns revolve around academic integrity and student performance, software faces technical (like code reliability and security) and legal risks (such as licensing and intellectual property).

Similar to Fu et al. (2024), in this study, PMT's risk-oriented framework with UTAUT's benefit-focused approach is combined to understand how software developers evaluate and respond to both the opportunities and risks posed by LLMs. Furthermore, it allows us to understand the underlying motivations that drive LLM adoption, such as performance benefits. In addition, it allows us to gain insights into how development teams evaluate and respond to risks such as code quality and security. Such a dual perspective can help us understand the complex decision-making process behind LLM adoption in software development environments.

3.2 Research framework

3.2.1 Hypothesis development

We recognize that employees are crucial stakeholders of any software development process and they incorporate LLMs in their projects (Ozkaya, 2023). Therefore, this study focuses on individual LLM adoption in software projects at the workplace.

To this extent, a new theoretical framework combining PMT and UTAUT is proposed. PMT is used to explain risk appraisals, coping appraisals, and maladaptive coping, while UTAUT to explain the benefit factors. Hypotheses are developed about factors influencing software developers' intention to use LLM outputs in their projects.

3.2.1.1 Maladaptive coping and behavioral intention

The maladaptive coping construct from the revised PMT theory represents avoidance of responses to perceived threats as opposed to addressing these threats (Rogers & Maddux, 1983). Maladaptive coping stems from stress and coping theory (Lazarus & Folkman, 1984) which describes how individuals assess and respond to stressful situations. Maladaptive coping is related to Emotion-Focused Coping, aimed at managing the emotional distress associated with the problem rather than solving it. Typical maladaptive responses include denial ("the risk is not real"), avoidance ("I prefer not to think about it"), or fatalism ("nothing I do can prevent it"). In the IT context, Liang and Xue (2009) use emotion-focused coping in their Technology Threat Avoidance Theory (TTAT) to explain different coping mechanisms for perceived threats. They define emotion-focused coping as a behavioral strategy oriented toward creating a false perception of the threat without in fact changing it, so that negative emotions related to threat are mitigated. Such behaviors typically manifest as avoidance, denial, wishful-thinking, and fatalism – responses that may provide temporary emotional relief but fail to address the actual threat (Milne et al., 2000).

Previous research applying PMT to technology adoption has shown that maladaptive coping behaviors can influence adoption intentions. For instance,

Chenoweth et al. (2009) found that maladaptive coping had a statistically significant effect on behavioral intention to adopt protective technologies. While their study focused on anti-spyware software, the underlying psychological mechanism of avoiding rather than addressing perceived threats is relevant to LLM adoption.

Similar to how users might avoid dealing with security threats by postponing anti-spyware installation (Chenoweth et al., 2009), software developers might cope with LLM risks through avoidance. PMT posits that maladaptive coping will positively influence behavioral intention. Therefore, I hypothesize:

H1: Maladaptive coping has a positive effect on the behavioral intention to adopt LLM in software projects.

3.2.1.2 Threat appraisal components

According to PMT, threat appraisal consists of two key components: perceived vulnerability and perceived severity (Rogers, 1975). Anderson and Agarwal (2010) highlight that the evolution of threats is related to individuals' attitudes and intentions to carry out security-related behaviors concerning the Internet and to one's computer (e.g., antivirus software, firewalls, system updates). When these threat perceptions are high, individuals can engage in either adaptive or maladaptive coping responses (Rogers & Maddux, 1983). Based on this, how these threat appraisal components affect both maladaptive coping and behavioral intentions is examined in the context of LLM adoption.

Maddux and Rogers (1983) demonstrated that when individuals perceive high vulnerability to threats, they may adopt different decision-making strategies - either a "precaution strategy" or a "hyperdefensiveness strategy." This is supported by the Technology Threat Avoidance Theory (TTAT), which suggests that when users

perceive threats as unavoidable, they are more likely to engage in emotion-focused coping (Liang & Xue, 2009). In the context of LLM adoption, Perceived Vulnerability is defined as the likelihood of using flawed LLM output in projects. The vulnerability or susceptibility to flawed outputs may lead developers to engage in maladaptive coping through avoidance instead of developing risk management strategies. Therefore, I hypothesize:

H2a: Perceived vulnerability is positively associated with maladaptive coping.

PMT has been used to show that higher perceived vulnerability increases protective behaviors. For instance, Alneyadi et al. (2023) found that users who viewed themselves as vulnerable to cyber threats were more likely to adopt AI-based cybersecurity systems. Similarly, Chenoweth et al. (2009) found that users who viewed themselves as vulnerable to spyware threats were more likely to adopt anti-spyware software.

Applied to LLM use in development teams, I hypothesize that developers could be less inclined to adopt LLMs or use LLM outputs when they view the likelihood of flawed outputs as high. Thus:

H2b: Perceived vulnerability has a negative effect on behavioral intention to adopt LLM in software projects

According to PMT's threat appraisal process, a high perceived severity of negative consequences can trigger an emotion-focused coping response when individuals feel unable to manage the threat (Rogers, 1975). For instance, they could resort to maladaptive coping strategies like avoidance or denial rather than engaging with the technology. Xin et al. (2021) find a correlation between perceived severity and hopelessness when examining individual users' responses to mobile malware threats. Liang and Xue (2009) predict users threat perception is determined by that

threat severity. When the user's treat perception is higher, they are more inclined to perform maladaptive coping with a focus on emotional distress.

In this study, I focus on the perceived severity of problems when employees use flawed LLM outputs in projects. In the LLM context, when developers perceive severe consequences from flawed outputs, they may resort to maladaptive coping mechanisms such as avoidance rather than implementing safeguards. Thus, I hypothesize:

H3a: Perceived severity has a positive effect on maladaptive coping

Liang and Xue (2009)'s theoretical framework posits that IT threat perception is determined by the perceived severity of malicious IT. They propose perceived severity has a positive interaction effect on perceived threat. When users perceive severe potential consequences from a technology's use, they are more likely to view it as threatening rather than beneficial.

Applied to our context of LLM use, perceived severity is the magnitude of potential negative consequences from using flawed LLM outputs, such as integrating inaccurate or legally problematic information or code into projects. Unlike protective technologies where perceived severity motivates adoption, the perceived severity of flawed LLM outputs could discourage LLM use. Therefore, I hypothesize:

H3b: Perceived severity has a negative effect on behavioral intention to use LLM in projects

3.2.1.3 Coping appraisal

Coping appraisal determines how individuals respond to technological threats. According to PMT (Rogers, 1975), after evaluating a threat, users engage in coping appraisal to assess their ability to respond to the threat. Liang and Xue (2009)

highlight that users who perceive a threat as avoidable are more likely to take safeguarding measures and, thus, are less likely to engage in maladaptive coping strategies. Conversely, when users perceive threats as unavoidable, they tend to incline to emotional coping. Other scholars support this observation, with Chen et al. (2022) demonstrating that perceived coping efficacy is the most influential element in both encouraging adaptive coping behaviors and discouraging maladaptive coping behaviors. Therefore, coping appraisal is essential for predicting whether users will respond adaptively or maladaptively to technological threats.

In PMT, response efficacy is part of the coping appraisal process. Response efficacy is the perceived effectiveness of a recommended coping response, and has emerged as one of the most consistent predictors of behavioral intentions in technology adoption (Tsai et al., 2016). In the context of this study, response efficacy refers to the belief that human oversight by critically evaluating LLM outputs for accuracy and legal compliance.

Prior research has demonstrated response efficacy's role in reducing maladaptive coping behaviors. Liang et al. (2019) conducted a study combining an experiment and a survey examining various ISec threats (phishing, malware, identity theft) and protective responses (antivirus software, strong passwords). Their findings showed that users who believed security threats could be avoided through protective measures were less likely to respond maladaptively through denial, wishful thinking and avoidance. Based on this theoretical foundation and empirical evidence, when developers believe in the effectiveness of human oversight as a protective measure against LLM-related risks, they may be less likely to resort to maladaptive coping strategies. Therefore:

H4a: Response efficacy has a negative effect on maladaptive coping

Tsai et al. (2016)'s research on ISec concerned with malware-related threats and security measures demonstrated that response efficacy predicts security behavior. Users believing in the effectiveness of security measures are more likely to adopt such measures. More recently, in the context of AI technologies, (Alneyadi et al., 2023) found that perceived response efficacy positively influenced users' intentions to adopt AI cybersecurity systems. Fu et al. (2024) demonstrated that response efficacy and self-efficacy influence the intention to use ChatGPT, and users felt confident in their ability to counteract potential negative outcomes.

Following this, I posit that when developers believe human oversight is effective for evaluating and verifying LLM outputs, they are more likely to use LLMs in their projects. Therefore:

H4b: Response efficacy has a positive effect on behavioral intention to use LLMs in projects.

In PMT, self-efficacy refers to one's confidence in their ability to successfully execute protective behaviors. PMT predicts that when individuals possess high self-efficacy, they are more likely to engage in protective behaviors rather than avoidance strategies. Conversely, those lacking confidence in their ability to implement protective measures often resort to emotion-focused coping and avoidance behaviors (Y. Chen & Zahedi, 2016). In this study's context, self-efficacy refers to users' confidence in their ability to critically evaluate LLM outputs to avoid using flawed content.

Yazdanmehr et al. (2023) found that self-efficacy negatively moderates the relationship between security related stress and emotion-focused coping. Thus, I hypothesize:

H5a: Self-efficacy has a negative effect on maladaptive coping

Researchers have also demonstrated self-efficacy's positive influence on the technology adoption intentions. In the context of AI, Alneyadi et al. (2023) found that users with higher confidence in their ability to use AI security systems were more likely to adopt these systems. This aligns with the Liang and Xue (2009)'s proposition that users are more likely to perceive threats as avoidable when they feel confident in implementing safeguarding measures. Similarly, Crossler (2010) demonstrated that self-efficacy positively affects protective behaviors like data backups. Therefore, I hypothesize:

H5b: Self-efficacy has a positive effect on behavioral intention to use LLM outputs in projects

According to PMT 2 (Rogers & Maddux, 1983), when individuals believe that taking preventive action requires a considerable amount of effort, time or resources, they are more likely to engage in maladaptive coping rather than implementing protective measures. These preventive actions therefore have a response cost for the individual. Maladaptive coping often manifests as denial, avoidance, or rationalization of inaction, and serves as a psychological mechanism to manage the anxiety induced by unaddressed threats.

This relationship has been empirically validated for technology adoption by (Chenoweth et al., 2009). They found that there is a significant positive relationship between response cost and maladaptive coping in the context of anti-spyware software adoption. In this study, response cost is the overhead associated with verifying LLM outputs. Drawing parallels to Chenoweth et al. (2009)'s findings, I hypothesize that when users perceive the cost of verifying LLM outputs as too high, they will be more likely to engage in maladaptive coping strategies.

H6a: Response cost has a positive effect on maladaptive coping

According to Liang and Xue (2009), users perceive an IT threat as avoidable and take a safeguarding measure when the measure is less costly. This relationship is supported by Zhang and McDowell (2009), who found that response efficacy and fear predicted the intention to use strong passwords, whereas response costs negatively influenced this intention. Similarly, Tsai et al. (2016) corroborated this observation by demonstrating that if security measures are perceived as difficult or time-consuming, users are less likely to adopt them. Applying this relationship to LLM usage where human oversight requires significant verification and evaluation effort, I hypothesize:

H6b: Response cost has a negative effect on behavioral intention to use LLM outputs in projects

3.2.1.4 Benefit components

While PMT provides insights into how users evaluate and respond to technology-related threats, LLM adoption is also driven by its perceived benefits. To understand adoption intentions, benefit-related constructs from UTAUT by (Venkatesh et al., 2003) are included in the model.

I focus on three components relevant to LLM adoption in software development: performance expectancy, task efficiency and effort expectancy. Performance expectancy and effort expectancy have been shown to influence attitudes toward technology adoption (Dwivedi et al., 2019; Khalilzadeh et al., 2017). Additionally, Fu et al. (2024) emphasized the importance of task efficiency in rationalizing ChatGPT use within users' workflows.

Performance expectancy is the extent to which individuals believe that using a technology will improve their job performance. It has been identified as a primary

determinant of technology adoption (Blut et al., 2022). In this study, it is defined as the degree to which employees believe that using LLM output helps them achieve higher productivity, accomplish more in less time, and ultimately enhance their professional success. Studies have demonstrated that when users perceive a technology as beneficial to their productivity and efficiency, they are more likely to adopt it (Dwivedi et al., 2019; Khalilzadeh et al., 2017).

The positive relationship between performance expectancy and intention to use is supported by various studies. A meta-analysis by Blut et al. (2022) confirms that performance expectancy remains the strongest predictor of behavioral intention to use technology. Fu et al. (2024) found that users in higher education contexts who perceived ChatGPT as enhancing their academic performance were more likely to use these tools in their work. Similarly, Camilleri (2024) validated that performance expectancy influences users' intentions to adopt LLMs. These findings suggest that when users believe LLMs will enhance their performance, they have stronger conviction to use these tools. Therefore:

H7: Performance expectancy has a positive effect on intention to use LLM outputs in projects

Recent research on LLMs has recognized task efficiency as a determiner of technology adoption, extending beyond UTAUT constructs (Fu et al., 2024; Hsu & Silalahi, 2024). Fu et al. (2024) investigated the benefit-risk-coping paradox of ChatGPT adoption in higher education, defining task efficiency as the extent to which the technology enables users to complete tasks more effectively and with reduced effort. Their study revealed that task efficiency influenced behavioral intention. This relationship was further validated by Fu et al. (2024) and Hsu and Silalahi (2024), who expanded the UTAUT framework by positioning task efficiency

as a motivating factor separate from performance expectancy. Their research showed that when users perceive AI tools as enhancing their task completion efficiency, they show stronger adoption intentions and increased actual usage behavior. Notably, task efficiency emerged as an even stronger predictor of technology use than effort expectancy in their study.

Building on these empirical findings, task efficiency is defined as the extent to which employees perceive that using LLM outputs enhances the efficiency of completing tasks. The evidence from these studies showing the significant positive relationship between perceived efficiency benefits and usage intention leads us to hypothesize:

H8: Task efficiency has a positive effect on intention to use LLM outputs on projects.

Effort expectancy, defined as ease of using a technology (Venkatesh et al., 2003), has been identified as a determiner of technology adoption. This is supported by the meta-analysis from Blut et al. (2022) Which shows that effort expectancy's impact is particularly pronounced for mobile technologies. Thus, when users perceive a technology as easy to use, they are more likely to develop an intention to use it. Similarly, Dwivedi et al. (2019) found in their meta-analysis that effort expectancy significantly influenced behavioral intention in over 162 articles they have covered. In the context of LLMs, users' perceptions of how easy it is to interact with and use LLM is likely to influence their adoption intentions. Therefore:

H9: Effort expectancy has a positive effect on intention to use LLM outputs on projects

3.2.1.5 Behavioral intention (dependent variable)

The UTAUT model proposed by (Venkatesh et al., 2003) establishes behavioral intention as a key determinant of actual technology use. Behavioral intention is defined as the user's motivation to engage with the technology and is considered the strongest predictor of actual system usage (Dwivedi et al., 2019). This finding is supported in longitudinal studies. For example, Blut et al. (2022)'s meta-analysis examined 192 empirical studies with a cumulative sample size of 67,497 users. They found that behavioral intention had a positive correlation with use. The UTAUT model further delineates four moderating variables influencing the strength of the relationship between behavioral intention and actual use.

Experience – The more experience a user has, the stronger the effect of behavioral intention on actual usage.

Voluntariness of use – In mandatory settings, social influence plays a role, whereas in voluntary settings, intention remains the dominant factor.

Gender – Intention of males to use technology is more strongly driven by performance expectancy, while females are influenced by effort expectancy and social influence.

Age – Younger users are more likely to be influenced by behavioral intention alone, whereas older users also rely on facilitating conditions.

In this research examining individual adoption of LLMs, I incorporate these moderating variables. However, voluntariness of use is excluded, following the precedent set by UTAUT 2 (Venkatesh et al., 2012), which eliminated this moderator due to its shift from organizational to consumer contexts. This adaptation aligns with our focus on individual-level adoption patterns.

In the context of this study, behavioral intention is employees' decision to use, stop using or continuing to use LLM output in projects. Drawing from both the meta-analyses and the UTAUT framework, I propose the following hypothesis:

H10: Behavioral intention has a positive effect on actual use of LLM outputs on projects

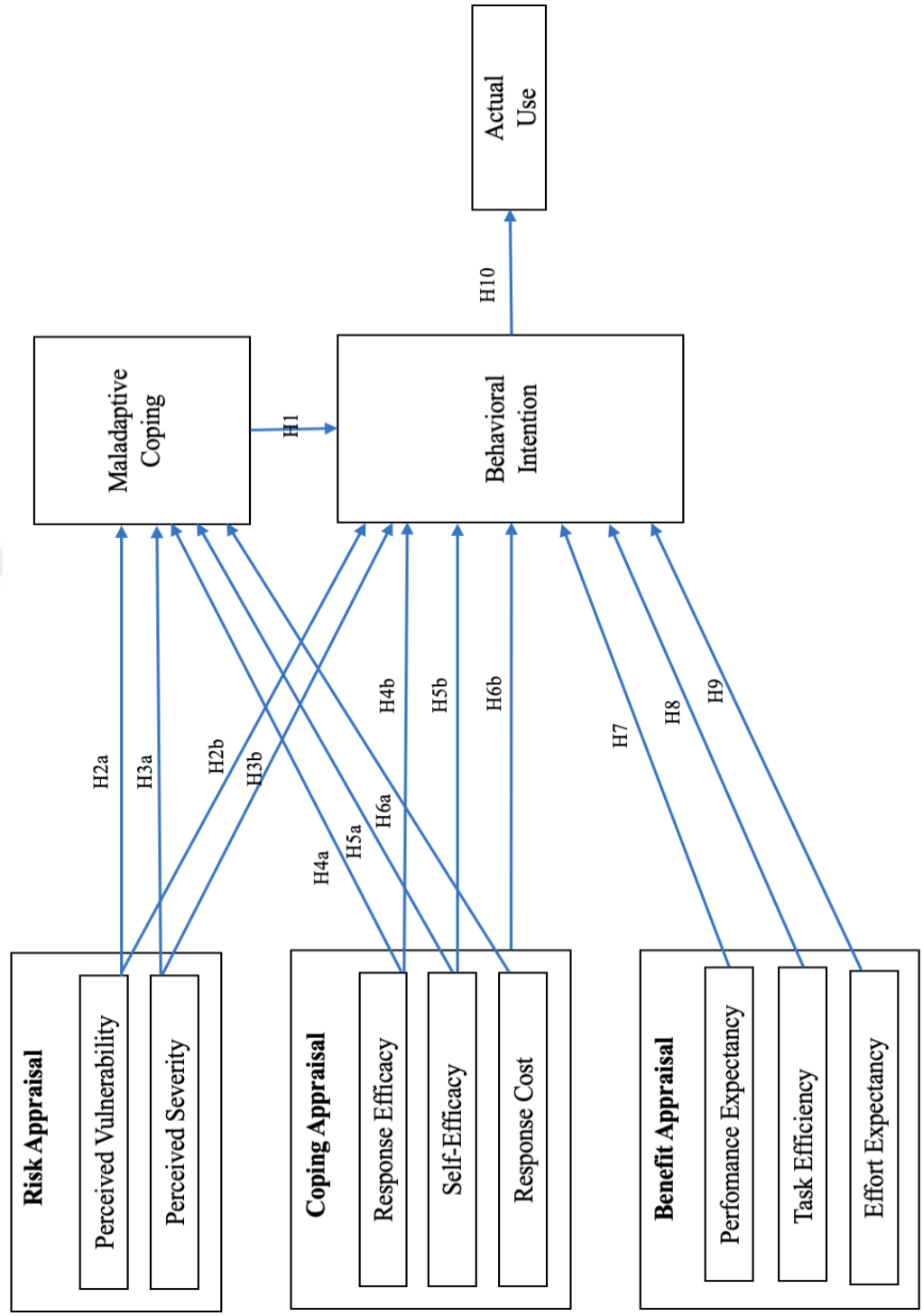

Figure 5 Hypothesized model

CHAPTER 4

METHODOLOGY

This chapter outlines the research methodology used to investigate LLM adoption in software development teams. It describes the research design, population and sampling approach, instrument development process and data collection process. The chapter details how the survey was developed, translated, and conducted to measure the theoretical constructs in the research model while establishing the methodological foundation for the subsequent analysis of factors influencing LLM adoption in software development environments.

4.1 Research design

This study adopts a positivist research paradigm which emphasizes objective measurements and statistical data analysis. Creswell and Creswell (2017) describe survey research as a methodology aimed at capturing numerical insights into trends, attitudes, or perspectives within a given population by analyzing data collected from a sample of that population. It includes cross-sectional and longitudinal studies using questionnaires or structured interviews for data collection with the intent of generalizing from a sample to a population. Based on the research questions, objectives and the foreseen analysis, the most suitable strategy for this study is a quantitative approach with data collected through questionnaire surveys. After the data is collected, the goal is to develop and quantitatively test the model that explains the adoption factors of LLMs in software development teams.

The unit of analysis in this research is singular team members working on software development projects, as the study investigates factors influencing LLM adoption at the individual level within software teams.

4.2 Population and sampling

The population for this study consists of individual software team members who have the potential to adopt LLMs in their workplace. To ensure a representative sample, inclusion criteria were established requiring that respondents be currently employed in software development or related roles where LLM adoption is relevant.

To ensure that respondents belonged to the appropriate sample frame, the online survey instrument started with an introduction explaining the purpose of the research. The introduction section of the survey instrument can be found in Appendix A.

The participants in the study included software developers, project/program managers, business analysts, quality assurance engineers, product owners, technical leads, data science specialists and other relevant roles.

In the Management Information Systems (MIS) literature, there is no universally agreed-upon minimum sample size, as different researchers have proposed varying thresholds depending on study context. Some suggest a minimum of five (Gopal et al., 1992) or ten times (Hair et al., 2011), of the hypotheses count. In our model 15 hypotheses were presented. Thus, 150 would be the minimum sample size required for this research. To determine an appropriate sample size, two methodologies were employed. The first methodology is the rule of thumb suggested by Hair et al. (2011). The second methodology followed is from Kock and Hadaya (2018). Gamma-exponential method was chosen due to its greater precision in

approximating true standard errors. The rule of thumb for the gamma-exponential method, the minimum absolute path coefficient should be 0.197, which corresponds to Cohen's f2 value of 0.04 —twice the minimum threshold. In this study, a significance level of 0.95 and a statistical power of 0.80 were satisfied. Based on these parameters, the minimum required sample size for the gamma-exponential method calculated to be 146. In this research, 151 survey participants have accumulated satisfying both limits.

The sampling approach was non-probability convenience sampling as the survey was distributed through professional networks via email and social media platforms including LinkedIn, WhatsApp, Telegram and BiP. Given the time and resource constraints, as well as the need to reach participants with relevant experience in technology use, this approach allowed reaching the minimum sample size.

4.3 Instrument development

4.3.1 Survey development process

The data collection tool used in this study was a structured questionnaire designed to measure the constructs defined in the research model. Following a comprehensive literature review about AI risks and AI adoption in MIS literature, factors are identified that have the potential to influence this adoption among software development teams. The survey instrument was developed by adapting measurement scales from UTAUT and PMT theories.

Prior to the main survey questions, two key concepts were explained to ensure consistent understanding among respondents:

1. Flawed Large Language Model (LLM) output: Defined as incorrect, biased or legally non-compliant content such as infringing on intellectual property rights, copyrights or sensitive data leakage. This term was introduced in this study to clearly delineate problematic AI outputs.
2. Human oversight on LLM output:
    a. Ability to detect errors in LLM outputs
    b. Knowledge of when to intervene
    c. Confidence in reviewing outputs for legal compliance

To establish a common understanding, the European Commission's definition of human oversight (*Article 14*, n.d.-b) is adapted.

The survey instrument also included demographic questions to gather information that might serve as moderating variables:

- Gender
- Age
- Education Level
- Work Experience
- Occupation
- Industry
- Location

4.3.2 Measurement scales

The survey instrument was developed by adapting measurement scales from existing literature in UTAUT and PMT. The questionnaire consisted of multiple indicators for each construct. This study is a reflective model and therefore all the constructs in this study were measured as reflective constructs. Each construct was measured using a 7-point Likert scale, as in the original articles, ranging from "Strongly Disagree" (1)

to "Strongly Agree" (7) to ensure sufficient variance in responses and to maintain consistency with the original scales.

Table 3 List of Constructs and Measurement

Construct	Measurement Type
Perceived Severity	Reflective
Perceived Vulnerability	Reflective
Self-Efficacy	Reflective
Response Efficacy	Reflective
Response Cost	Reflective
Performance Expectancy	Reflective
Effort Expectancy	Reflective
Task Efficiency	Reflective
Behavioral Intention	Reflective
Actual Use	Reflective

For several constructs, including Perceived Severity, Response Efficacy, Self-Efficacy, and Behavioral Intention, I deliberately collected items from multiple established sources. This approach provided alternative measurement options for these constructs. Consistent with best practices in PLS-SEM, the statistical properties of all candidate indicators are evaluated and retained those that demonstrated stronger outer loadings, improved discriminant validity, and theoretical coherence (Hair et al., 2022). For instance, when multiple indicators were available for a single construct, the ones that provided better discriminant validity, stronger path relationships, and more precise representation of the theoretical construct within this research context. This methodical selection process enhanced and ensured the overall model quality by allowing us to utilize the best fit measurement items for each construct rather than being limited to a single source of items that might not optimally capture the constructs in the LLM adoption context. Indicators and the

references can be found at Appendix B and the Likert scales for each indicator can be found in Appendix C.

4.3.3 Translation process

As the original measurement scales were in English and the target respondents were mainly Turkish-speaking professionals, a rigorous translation process was implemented. The questionnaire items were first translated into Turkish. To ensure conceptual equivalence, and eliminate any inconsistencies the Turkish version was then back-translated to English as suggested by Brislin (1970).

After translations were done, feedback was collected from subject matter experts to verify whether the items were clear and whether the translation had affected the meaning of the items. Minor adjustments were made based on this feedback.

4.4 Data collection, preparation procedures

Data for this study were collected between 21 February 2025 and 11 April 2025 using a self-administered online questionnaire. The study received ethical approval from the Boğaziçi University ethics committee prior to data collection (see APPENDIX E). Participation in the questionnaire was entirely anonymous and voluntary. The questionnaire was prepared in both English and Turkish, accommodating international and local contacts as well as colleagues. Participants could choose their preferred language.

The survey was distributed via multiple digital channels, including professional networks and peer-to-peer (P2P) messaging platforms. Specifically, it was shared in

developer-focused groups and chats on WhatsApp, Telegram and BiP, as well as through direct messages and a public post on the researcher's LinkedIn profile.

A total of 151 complete responses were collected through this distribution process. All survey items were marked as mandatory, which eliminated the possibility of partial responses. The data were downloaded in .xlsx format via Google Forms and subsequently prepared for analysis.

Following data collection, several steps were taken to clean and prepare the dataset for analysis. First, responses from the English and Turkish versions of the survey were consolidated. Since both versions were distributed using separate Google Forms, their respective .xlsx files were manually merged into a single dataset. During this process, the Turkish responses to descriptive questions (such as gender, occupation and education) were replaced with their English versions.

All descriptive demographic variables (e.g., gender, age, education level, work experience, occupation, industry, country, and actual use) were recoded numerically. For example, gender responses were coded as 1 = Female, 2 = Male, and 3 = Prefer not to say. Similar coding schemes were applied across all other demographic variables to enable statistical analysis.

In addition, the survey indicators were manually labelled using the construct abbreviations. For instance, PS1, PS2, PS3 and so on for Perceived Severity's indicators, and PV1 and PV2 for Perceived Vulnerability, and so on. These labels were retained to maintain clarity and with the theoretical model.

Finally, care was taken to remove any non-ASCII characters from the dataset to ensure compatibility with the statistical software and avoid processing errors.

CHAPTER 5

ANALYSIS

This chapter explains the analysis of the study. The chapter starts with presenting the demographic profile of respondents, and continues with the analytical approach, as well as measurement and structural model assessments.

5.1 Demographic profile of respondents

A total of 151 complete responses were collected through the online questionnaire from individuals working in software development or related fields where LLM adoption is relevant.

Table 4 presents the detailed demographic profile of survey respondents. In terms of gender distribution, the sample included 112 males (74.2%) and 37 females (24.5%), and two people preferred not to say (1.3%). The age distribution shows that the majority of respondents were between 25-29 years (53 respondents, 35.1%), followed by the 30-35 age group (40 respondents, 26.5%). For work experience data, professionals between three to 10 years of total work experience, accounted for 46.2% of the sample, representing nearly half of all participants.

Table 4 Demographics of Survey Respondents

Category	Freq	%	Category	Freq	%
Gender			Work Experience		
Male	112	74.2	Less than 3 years	35	23.2
Female	37	24.5	3 years or more, but less than 5 years	32	21.7
Prefer not to say	2	1.3	5 years or more, but less than 10 years	37	24.5
Age			10 years or more, but less than 15 years	28	18.5
< 22 years	2	1.3	15 years or more	19	12.6
22-25 years	24	15.9	Occupation		
25-29 years	53	35.1	Software Developer (Backend, Frontend, Full Stack, etc.)	109	72.2
30-35 years	40	26.5	Business Analyst	13	8.6
36-41 years	23	15.2	Technical Lead	10	6.6
42-50	7	4.6	Data Science Specialist	6	3.9
51-60	2	1.3	Product Owner	5	3.3
Education Level			Project / Program Manager	5	3.3
High school graduate and less	2	1.4	Solution Architect / System Architect	3	1.9
University student	10	6.6			
Graduated from a university	94	62.3			
Master's/PhD student/graduate	45	29.8			

The majority of respondents were software developers (109 respondents, 72.19%), followed by business analysts (13 respondents, 8.61%) and technical leads (10 respondents, 6.62%). Other roles included project / program managers, data science specialists and product owners, solution architects or system architects are each less than 4% of the participants. The predominance of software developers in the sample mirrors current research trends, where the majority of academic literature focuses specifically on LLM applications in coding and development tasks rather than on pre-development planning or post-development activities (Fan et al., 2023; Hou et al., 2024). This concentration of developers in the sample provides valuable insights into LLM adoption among the professional group most directly affected by these technologies in their daily work.

5.1.1 LLM usage frequency

The respondents were asked about their actual use of LLM tools within their projects. Table 5 presents the LLM usage frequency among respondents. The data reveal that a substantial number of respondents (41.1%) use LLM tools multiple times daily which suggests a deep reliance on these technologies in development workflows. Combined with participants who use LLMs once daily (4%), nearly half of all participants engage with LLMs on a daily basis. Notably, 14.6% of respondents reported never using LLM tools in their projects, suggesting that despite the growing popularity of these technologies, a significant portion of the software development professionals have yet to adopt them. This distribution provides a baseline for general LLM penetration in professional software development settings and helps contextualize the subsequent analysis of adoption factors.

Table 5 LLM Usage Frequency Among Respondents

LLM Usage Frequency	Frequency	Percentage
Several times a day	62	41.1%
Several times a week	29	19.2%
Never	22	14.6%
Several times a month	19	12.6%
Once a month	7	4.6%
Once a day	6	4.0%
Once a week	6	4.0%

5.1.2 Industry and geographic distribution

I asked respondents about the industry their company operates in and their company's country of operation. This demographic information helps understand LLM adoption across different sectors and geographical regions. It can potentially reveal industry-specific or regional patterns in adoption behaviors.

Table 6 presents the distribution of respondents across industries. The majority of participants work in Technology, Media and Telecommunications (78 respondents, 61.9%), followed by Financial Services (28 respondents, 22.2%). The remaining respondents are distributed across various sectors.

Table 6 Industries

Industry	Frequency	Percentage
Technology, Media and Telecommunications	94	62.3
Financial Services	29	19.2
Transportation and Logistics	6	4.0
Government Policy	5	3.3
Energy, Infrastructure and Natural Resources	4	2.6
Automotive	2	1.3
Tourism	2	1.3
Industrial Production	2	1.3
Retail and Consumer Products	2	1.3
Health	2	1.3
Pharmaceuticals and Life Sciences	1	0.7
Construction and Engineering	1	0.7
Private Equity	1	0.7

Table 7 shows the geographical distribution of respondents' companies. The majority of participants were based in Türkiye (104 respondents, 68.9%), followed by Belarus (13 respondents, 8.6%). This distribution diverges from the global patterns observed in larger-scale developer surveys. The most recent and the biggest developer survey was conducted by Stack Overflow Developer Survey in 2024. This disproportionate representation is likely attributable to the distribution method of the current survey, as it was distributed through professional and academic networks primarily accessible to the researcher.

Table 7 Demographics About Countries

	Categorization	Frequency	Percent
Country	Türkiye	104	68.9
	Belarus	13	8.6
	France	6	4.0
	Germany	5	3.3
	USA	5	3.3
	Spain	4	2.6
	United Kingdom	3	2.0
	India	3	2.0
	Estonia	2	1.3
	Finland	1	0.7
	Sweden	1	0.7
	Italy	1	0.7
	Russia	1	0.7
	Qatar	1	0.7
	United Arab Emirates	1	0.7

To check for potential nonresponse bias, early and late survey respondents were compared, following the approach of Armstrong and Overton (1977). Using response timestamps, the full sample was split into early and late respondents, and independent sample t-tests were performed on demographic items. Most items showed no significant differences between the two groups. Two items (Age and Experience) showed significant differences, the effect sizes were small (Lakens, 2013). These results show that nonresponse bias is not likely to be a concern in this study.

5.2 Analysis of the model

This study employed Partial Least Squares Structural Equation Modelling (PLS-SEM) to estimate and validate the proposed research model. PLS, which is an SEM-based tool, is a variance-based, nonparametric technique well-suited for prediction-focused research. It accommodates both formative and reflective measurement models and handles linear as well as nonlinear relationships, even in complex models (Chin & Newsted, 1999). The decision to apply PLS-SEM is grounded in both the study's objectives and its design characteristics, in alignment with recommendations from Urbach and Ahlemann (2010). This exploratory research study has introduced a new conceptual model grounded in two established theories, UTAUT and PMT. This approach differs from developing an entirely new theory or merely validating an existing one; rather, it involves restructuring established theoretical frameworks in a novel context, LLM adoption. PLS-SEM is particularly suited for such model development studies. Additionally, the relatively limited sample size in this research, fewer than 200 participants, aligns with PLS-SEM's suitability for small to medium samples (Urbach & Ahlemann, 2010). The non-normal distribution characteristics of the responses support using PLS-SEM. The study aims to explain the variance in behavioral intention and actual LLM use by examining the influence of various factors. Accordingly, the primary focus is on identifying the relationships between constructs and the strength of those relationships. Finally, PLS-SEM has superior capacity to tolerate potential multicollinearity issues between predictor variables, which is important for our model that combines variables from different theoretical frameworks (Hair, Ringle, et al., 2011).

In this study, WarpPLS 8.0, a PLS-SEM-based statistical tool, is used to conduct the analysis. WarpPLS provides the estimated path coefficients and allows

for the simultaneous assessment of both the measurement and structural models (Chin et al., 2003). WarpPLS can estimate both linear and nonlinear relationships among latent variables without requiring manual specification. It provides a comprehensive set of models fit and quality indices — including the Average Path Coefficient (APC), Average R-squared (ARS), and Average Full Collinearity VIF (AFVIF) — making it suitable for evaluating both the structural and measurement models.

5.2.1 Measurement model assessment

The measurement model was evaluated using standard validity and reliability metrics recommended in PLS-SEM (Urbach & Ahlemann, 2010). First, the reliability metrics of the model are checked. Internal consistency reliability is typically evaluated using Cronbach's alpha (CA), which has long been the standard measurement approach. As a complementary assessment method, researchers also employ composite reliability (CR) (Werts et al., 1974). Both metrics serve to determine the reliability of measurement constructs, though they approach this evaluation from slightly different methodological perspectives. While CA assumes that all indicators contribute equally to the construct, CR takes into account that indicators have different loadings (Henseler et al., 2009).

Regardless of the coefficients used for internal consistency, values above 0.7 are desirable. Convergent validity was established through Average Variances Extracted (AVE). Values exceeding 0.5 threshold indicates adequate convergent validity (Urbach & Ahlemann, 2010). Additionally, loadings and cross-loading analysis are used for confirming convergent validity which will be discussed in greater detail in the section addressing indicator reliability.

Table 8 below shows that both CR and CA values exceed 0.7, and AVE values are greater than 0.5, which satisfy the thresholds.

Table 8 Composite Reliability Coefficients and Cronbach's Alpha Results for Internal Consistency Coefficients AVE for Convergent Validity

	Composite Reliability Coefficients	Cronbach's Alpha Coefficients	Average variances extracted
PS	0.909	0.849	0.768
PV	0.921	0.870	0.795
RE	0.901	0.835	0.753
SE	0.922	0.872	0.797
RC	0.848	0.727	0.653
PE	0.941	0.875	0.889
EE	0.953	0.935	0.836
TE	0.940	0.904	0.839
MC	0.906	0.844	0.763
BI	0.955	0.929	0.876
AU	1.000	1.000	1.000

To verify discriminant validity, I examined whether each construct's square root of AVE exceeded its correlations with other latent variables. As demonstrated in Table 9, all constructs met this criterion, confirming discriminant validity in our measurement model (Kock & Lynn, 2012).

Table 9 Correlations Among Latent Variables and Square Roots of Average Variance Extracted Values

	PS	PV	RE	SE	RC	PE	EE	TE	MC	BI	AU
PS	0.877										
PV	-0.143	0.891									
RE	0.171	-0.150	0.868								
SE	0.037	-0.128	0.516	0.893							
RC	0.387	0.223	-0.153	-0.075	0.808						
PE	0.055	0.047	0.429	0.485	-0.004	0.943					
EE	0.139	-0.138	0.473	0.698	-0.071	0.626	0.915				
TE	0.010	0.132	0.452	0.463	0.006	0.765	0.554	0.916			
MC	0.010	0.374	-0.147	-0.125	0.248	0.078	-0.078	0.130	0.873		
BI	0.020	0.024	0.450	0.578	-0.001	0.731	0.710	0.670	0.093	0.936	
AU	-0.067	0.010	0.308	0.442	-0.174	0.514	0.528	0.500	-0.154	0.517	1.000

Discriminant validity was further assessed using the Heterotrait-Monotrait (HTMT) ratio of correlations. HTMT values are considered to be good if < 0.9 and best if < 0.85. As shown in Table 10, the HTMT values for most construct pairs were well below the threshold of 0.85, indicating discriminant validity between those constructs. Only the relationship between Performance Expectancy (PE) and Technology Enhancement (TE) showed a slightly elevated HTMT value of 0.86. (Henseler et al., 2015) recommend a threshold of 0.9 for conceptually similar constructs and 0.85 for conceptually distinct constructs. Therefore, 0.9 is the recommended limit.

Table 10 HTMT Ratios

	PS	PV	RE	SE	RC	PE	EE	TE	MC	BI	AU
PS											
PV	0.165										
RE	0.207	0.175									
SE	0.070	0.151	0.603								
RC	0.479	0.296	0.211	0.106							
PE	0.065	0.056	0.504	0.554	0.089						
EE	0.156	0.152	0.537	0.774	0.108	0.694					
TE	0.077	0.152	0.521	0.522	0.069	0.860	0.604				
MC	0.060	0.435	0.177	0.144	0.330	0.091	0.103	0.155			
BI	0.075	0.045	0.511	0.641	0.039	0.810	0.763	0.730	0.111		
AU											

Indicator reliability can be observed from cross-loading tables and was assessed to ensure that each item strongly represented its intended latent construct and did not significantly load on others (Hair, Ringle, et al., 2011). All indicators showed loadings higher than 0.7 for their intended constructs while showing lower values for other constructs in the cross-loadings table, with statistical significance at $p < 0.001$ (Amora, 2021). The cross-loading table can be found in Appendix D.

Additionally, the difference between each indicator's highest loading and its second-highest cross-loading was calculated to ensure that this difference exceeded 0.20 (Howard, 2016). This criterion supports discriminant validity by confirming that each item loads substantially higher on its associated construct than on any other construct.

Table 11 Cross-loading Control

	Max Loadings	2ndMax Loadings	Difference	If Dif > .20
PS4	0.897	0.096	0.801	TRUE
PS5	0.893	0.091	0.802	TRUE
PS6	0.839	0.113	0.726	TRUE
PV1	0.898	0.082	0.816	TRUE
PV2	0.915	0.123	0.792	TRUE
PV3	0.86	0.16	0.7	TRUE
RE1	0.851	0.209	0.642	TRUE
RE2	0.835	0.114	0.721	TRUE
RE3	0.915	0.096	0.819	TRUE
SE1	0.896	0.14	0.756	TRUE
SE2	0.926	0.126	0.8	TRUE
SE3	0.854	0.281	0.573	TRUE
RC1	0.851	0.146	0.705	TRUE
RC2	0.893	0.152	0.741	TRUE
RC3	0.663	0.247	0.416	TRUE
PE2	0.943	0.151	0.792	TRUE
PE3	0.943	0.129	0.814	TRUE
TE1	0.889	0.357	0.532	TRUE
TE2	0.93	0.075	0.855	TRUE
TE3	0.934	0.062	0.872	TRUE
EE1	0.904	0.051	0.853	TRUE
EE2	0.909	0.238	0.671	TRUE
EE3	0.937	0.059	0.878	TRUE
EE4	0.902	0.095	0.807	TRUE
MC1	0.826	0.331	0.495	TRUE
MC2	0.901	0.138	0.763	TRUE
MC3	0.89	0.231	0.659	TRUE
BI5	0.93	0.144	0.786	TRUE
BI6	0.955	0.056	0.899	TRUE
BI7	0.922	0.063	0.859	TRUE
Actual_Use	1000	0	1000	TRUE

To address potential common method bias (CMB) that might arise from online questionnaire data collection, Harman's single factor test and full collinearity variance inflation factors (FVIF) was conducted (Kock, 2015, 2021). By consolidating all indicators into a single latent variable and calculating the Average Variance Extracted (AVE) using WarpPLS. Following (Kock, 2021) guidelines, an AVE value below 0.5 for this aggregated latent variable indicates the absence of significant common method bias.

In Harman's single factor test, the AVE result was 0.332, which shows that there is no CMB concern in the indicators. VIF for each construct and the Average full collinearity VIF (AFVIF) values was also checked. According to (Kock, 2015), a VIF lower than 5 is good and 3.3 is ideal. In this research, each of the constructs is below 3.3, demonstrating that common method bias was not considered an issue for our model.

Table 12 Full collinearity VIFs

PS	PV	RE	SE	RC	PE	EE	TE	MC	BI	AU
1.406	1.330	1.689	2.233	1.442	3.180	3.086	2.863	1.337	3.101	1.716

5.2.2 Structural model assessment

To evaluate the relationships among latent variables and to test the proposed hypotheses, the structural model was assessed following the guidelines of PLS-SEM (Hair, Ringle, et al., 2011; Henseler et al., 2009). This stage of the analysis focuses on estimating the effect, direction, and significance of the hypothesized paths among constructs in the conceptual model.

In this analysis, I evaluated multiple models using the specified indicators. The model demonstrating the highest R^2 value was selected as our final model. As

shown in Table 13, the accepted model's performance metrics—including the average path coefficient (APC), Average R-squared (ARS), Average adjusted R-squared (AARS), and average full collinearity VIF (AVIF) values — all fell within expected parameters, confirming the model's statistical validity (Kock, 2024).

Table 13 Model Fit and Quality Indices

Average path coefficient (APC)	0.169, P<0.001
Average R-squared (ARS)	0.431, P<0.001
Average adjusted R-squared (AARS)	0.409, P<0.001
Average block VIF (AVIF)	1.622, acceptable if <= 5, ideally <= 3.3
Average full collinearity VIF (AFVIF)	2.059, acceptable if <= 5, ideally <= 3.3

P-values were calculated using the Stable3 method, which is the default nonparametric resampling approach in WarpPLS (Kock, 2024). For robustness checks, additional analyses using bootstrapping and jackknifing methods were also conducted. Bootstrapping creates many new samples by randomly picking from the data and jackknifing leaves out one participant at a time from the data (Urbach & Ahlemann, 2010). Stable3 is the recommended method in WarpPLS as it yields estimates that are more closely approximate actual standard errors (Kock, 2018). In addition, given that Stable3 produced the greatest number of statistically significant constructs ($p < 0.05$), it was selected as the main resampling approach for this analysis.

Table 14 P-values of Path Coefficients Using Stable3, Jackknifing and Bootstrapping Techniques

Stable3											
	PS	PV	RE	SE	RC	PE	EE	TE	MC	BI	AU
MC	0.017	<0.001	0.184	0.182	<0.001						
BI	0.295	0.249	0.394	0.105	0.491	<0.001	<0.001	0.016	0.227		
AU										<0.001	
Jackknifing											
	PS	PV	RE	SE	RC	PE	TE	EE	MC	BI	AU
MC	0.458	<0.001	0.204	0.271	0.005						
BI	0.369	0.428	0.377	0.132	0.490	0.003	0.068	0.006	0.120		
AU										<0.001	
Bootstrapping											
	PS	PV	RE	SE	RC	PE	TE	EE	MC	BI	AU
MC	0.264	<0.001	0.242	0.247	0.009						
BI	0.288	0.276	0.342	0.096	0.498	0.002	0.049	0.001	0.119		
AU										<0.001	

Path coefficients, which represent the strength and direction of relationships between the constructs in the theoretical model, its p-value reveal whether the hypothesized relationships were statistically significant. Table 15 presents the path coefficients with their corresponding p-values, and effect sizes. Results indicate that Perceived Severity (PS) has a positive influence on Maladaptive Coping (MC) with a path coefficient of 0.167 ($p = 0.017$) and a small effect size of 0.025. Similarly, Perceived Vulnerability (PV) demonstrates an impact on MC with a coefficient of 0.282 ($p < 0.001$) and a medium effect size of 0.130, while Response Cost (RC) also

significantly influences MC (β = 0.261, p < 0.001, effect size = 0.104). Self-efficacy and Response-efficacy has no significant effect on MC.

With regard to Behavioral Intention (BI), the analysis identified three significant determinants: Performance Expectancy (PE) positively affects BI with a coefficient of 0.353 (p < 0.001) and a high effect size of 0.265, Effort Expectancy (EE) exhibits a strong positive influence on BI (β = 0.299, p < 0.001, effect size = 0.213), and Task Efficiency (TE) also significantly impact on BI (β = 0.170, p = 0.016, effect size = 0.118). Furthermore, the model confirmed that BI significantly predicts Actual Use (AU) with a robust path coefficient of 0.527 (p < 0.001) and a large effect size of 0.278.

On the other side, perceived severity (PS), perceived vulnerability (PV), self-efficacy (SE), response efficacy (RE) and response cost (RC) had no significant influence on BI.

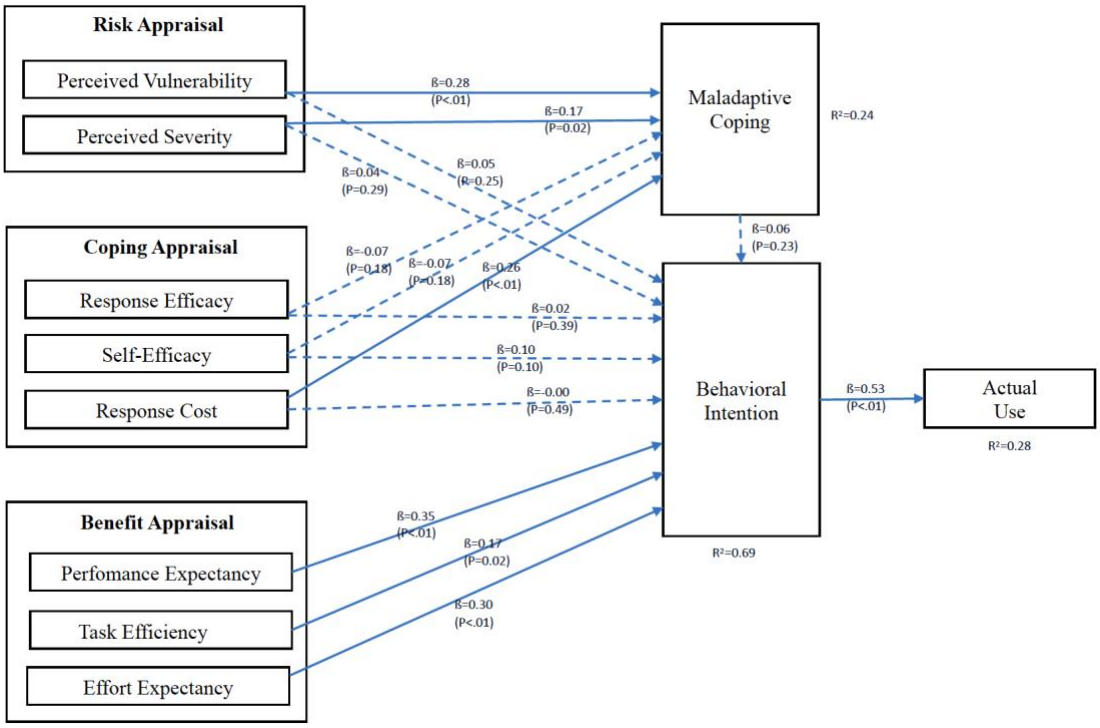

Figure 6 Model results

The results indicate that, there is no moderating factor detected that has a significant influence on actual use of LLMs.

Table 15 Path Coefficients, P values, and Effect Sizes

Path	Path Coefficients (ß)	P-value	Significance	Effect Size
PS → MC	0.167	0.017	Significant	0.025
PS → BI	0.044	0.295	Not significant	0.003
PV → MC	0.282	<0.001	Significant	0.108
PV → BI	0.055	0.249	Not significant	0.013
RE → MC	-0.072	0.184	Not significant	0.011
RE → BI	0.022	0.394	Not significant	0.010
SE → MC	-0.073	0.182	Not significant	0.009
SE → BI	0.100	0.105	Not significant	0.058
RC → MC	0.261	<0.001	Significant	0.084
RC → BI	-0.02	0.491	Not significant	0.001
PE → BI	0.353	<0.001	Significant	0.265
EE → BI	0.299	<0.001	Significant	0.213
TE → BI	0.170	0.016	Significant	0.118
MC → BI	0.060	0.227	Not significant	0.012
BI → AU	0.527	<0.001	Significant	0.278

The results of the questionnaire for the structural model are shown in Table 16. Seven out of fifteen hypotheses are supported. The significance of relationships between constructs is demonstrated through their p-values, effect sizes, and magnitudes.

Table 16 Hypotheses in the Model

Hypothesis	Supported / Not Supported
H1: Maladaptive coping has a positive effect on the behavioral intention to adopt LLM in software projects.	Not Supported
H2a: Perceived vulnerability is positively associated with maladaptive coping.	Supported
H2b: Perceived vulnerability has a negative effect on behavioral intention to adopt LLM in software projects	Not Supported
H3a: Perceived severity has a positive effect on maladaptive coping	Supported
H3b: Perceived severity has a negative effect on behavioral intention to use LLM in projects	Not Supported
H4a: Response efficacy has a negative effect on maladaptive coping	Not Supported
H4b: Response efficacy has a positive effect on behavioral intention to use LLMs in projects.	Not Supported
H5a: Self-efficacy has a negative effect on maladaptive coping	Not Supported
H5b: Self-efficacy has a positive effect on behavioral intention to use LLM outputs in projects	Not Supported
H6a: Response cost has a positive effect on maladaptive coping	Supported
H6b: Response cost has a negative effect on behavioral intention to use LLM outputs in projects	Not Supported
H7: Performance expectancy has a positive effect on intention to use LLM outputs in projects	Supported
H8: Task efficiency has a positive effect on intention to use LLM outputs on projects.	Supported
H9: Effort expectancy has a positive effect on intention to use LLM outputs on projects	Supported
H10: Behavioral intention has a positive effect on actual use of LLM outputs on projects	Supported

CHAPTER 6

DISCUSSIONS, LIMITATIONS, AND CONCLUSION

This chapter interprets the key findings from the structural equation modeling analysis and examines the relationships between constructs in the research model. This analysis takes into account methodological limitations and contextual factors. The chapter concludes by summarizing the research implications and proposing directions for future studies.

6.1 Factors affecting LLM adoption in software development teams

The analysis revealed that three UTAUT-derived constructs influence behavioral intention to use LLMs in the software development context, notably performance expectancy, effort expectancy and task efficiency.

The strongest indicator of behavioral intention was performance expectancy which indicates that software professionals' perception of productivity gains increases their intention to adopt LLMs. This aligns with the literature which identified performance expectation as the most important determinant for the intention to use a new technology (Blut et al., 2022). It is also corroborated by Camilleri (2024)’s finding of performance expectancy as a predictor for LLM adoption. Similarly, Fu et al. (2024) highlighted that clear performance gains significantly influence the intention to use ChatGPT. These findings, aligned with our study, shows that performance expectancy retains its explanatory power in new technologies– LLMs, with different user profiles – software professionals.

Effort expectancy exhibited a positive influence on behavioral intention in this study. This finding reinforces UTAUT explanatory power which recognizes

effort expectancy as a key predictor of adoption (Blut et al., 2022; Dwivedi et al., 2019). In the context of LLMs, Fu et al. (2024) reframe effort expectancy as the user's perception of convenience, clarity, and minimal cognitive load when interacting with ChatGPT. Their findings aligned with Hsu and Silalahi (2024) showed that such perceptions significantly influenced adoption decisions. Thus, the perceived ease of using LLMs impacts software developers' willingness to adopt these tools into their workflow. These findings collectively suggest that user-friendly interfaces and ease of implementation of LLM tools are critical for adoption.

Task efficiency was also found to have a significant effect on behavioral intention, suggesting that software developers are motivated to adopt LLMs when these tools enhance the effectiveness of task executions. This effect appears to extend beyond traditional UTAUT constructs, indicating that efficiency at the task level is a distinct and significant driver of adoption. Our findings align with Hsu and Silalahi (2024) and who also identified task efficiency as a standalone motivational factor beyond traditional UTAUT constructs which significantly influence AI adoption. Software professionals benefit from LLMs in code generation, debugging, and documentation (M. Chen et al., 2021a; Dvivedi et al., 2024; Tian et al., 2024). As these are essential stages of the SDLC process, efficiency improvements result in reduced overall development time. As shown by the analysis results, LLMs are well-suited for software development's task-oriented nature whereby improvements in task-level productivity increases the intention to adopt this technology.

These findings combined show that software professionals place importance primarily on the benefits of a new technology when considering its adoption. The effect sizes associated with these relationships highlight the explanatory power of the

UTAUT framework in understanding LLM adoption within the software development context.

Our analysis of the PMT constructs yielded insights regarding the threat appraisal process in LLM adoption. Perceived severity (PS) demonstrated a positive influence on maladaptive coping which suggests that developers' security concerns may increase avoidance or denial strategies when confronting risks of LLM use. This finding aligns with Rogers (1975)'s PMT model where a high perceived severity of negative consequences can trigger an emotion-focused coping response when individuals feel unable to manage the threat. Similar patterns also found in different studies such as Xin et al. (2021) found that high perceived severity in the context of mobile malware was associated with hopelessness and withdrawal, especially when users lacked the confidence to address the threat. Liang and Xue (2009) suggested that when perceived severity is high and users do not perceive adequate control or coping options, it may result in psychological stress and maladaptive behaviors. While the effect size of perceived severity is relatively small, this study also supports this relationship. This indicates that the awareness of LLMs' risks and their severity can lead to maladaptive coping mechanisms among software professionals such as avoidance or inaction.

Perceived vulnerability (PV) emerged as a stronger predictor of maladaptive coping in our model. This shows that developers who perceive a higher likelihood of encountering flawed LLM outputs tend to engage more in avoidance behaviors rather than adaptive risk management. This finding is consistent with PMT's (Rogers, 1975; Rogers & Maddux, 1983) suggestion. This underscores how perceived vulnerability to LLM-related risks can trigger psychological defense mechanisms. A similar pattern has been observed through other research techniques, notably the

Technology Threat Avoidance Theory (TTAT) framework developed by Liang and Xue (2009). TTAT is based on PMT and posits that when users perceive threats as likely and unavoidable, they are more inclined to adopt emotion-focused responses, such as avoidance or disengagement.

Similar to PS and PV, Response cost (RC) also demonstrated a significant positive effect on maladaptive coping as assumed by PMT. This indicates that the perceived overhead costs associated with implementing human oversight on LLM outputs contributes significantly to maladaptive coping strategies. When software professionals view the effort required for proper LLM output verification as excessive, they are more likely to adopt avoidance strategies rather than implementing appropriate safeguards. This finding is in line with Chenoweth et al. (2009) who found a positive link between response cost and avoidance in the context of anti-spyware software. In the literature review part, a series of LLMs' flawed output is highlighted such as hallucinations, biased outputs or security vulnerabilities, and their verification process can lead to cognitive overload or rigorous amount of time. Therefore, the perception of high implementation costs for risk mitigation strategies are likely to end in maladaptive behaviors.

Our findings revealed that the PMT-derived threat appraisal constructs (perceived severity and perceived vulnerability) and coping appraisal constructs (response efficacy, self-efficacy, and response cost) did not directly influence the behavioral intention to use LLMs. Several explanations may account for these unexpected findings, which we discuss below.

We hypothesized that maladaptive coping would positively impact behavioral intention to use LLMs, which was not supported by the results. In classical PMT models, maladaptive coping responses are thought to undermine protective actions.

For instance, someone in denial about a risk is less likely to adopt a recommended safeguard but such relationships are also context-dependent as in anti-spyware or health (Rippetoe & Rogers, 1987). In this study, the lack of a significant relationship can be caused by developers exhibiting maladaptive coping who still showed interest in LLMs due to other motivations. LLMs offer many tangible benefits as we discussed in the literature review part such as code generation (Jiang et al., 2024), debugging (Tian et al., 2024) and assistance in requirement elicitation (Cosler et al., 2023). These positive expectations may have overridden risk-based concerns. Thus, an individual who rationalizes that LLMs sometimes hallucinate might still intend to adopt LLMs because of perceived benefit gains. Kuhail et al. (2024) support this argument in their research as they found that developers who felt most threatened by AI were also frequent users due to the benefit gains. Other possible reasons could be AI integrations into development tools whereby developers receive automated coding assistance without explicitly opting in, as well as self-inflicted pressure to use AI tools to stay competitive.

The hypothesis that threat appraisal factors, perceived vulnerability and perceived severity negatively influences software professional's intention to adopt LLMs was not supported. In this study, perceived severity refers to the belief about the seriousness of potential harm resulting from flawed LLM outputs in software projects, while perceived vulnerability reflects the perceived likelihood of encountering such flawed outputs. These constructs were included to capture software professionals' risk perceptions. A likely explanation that our findings do not align with previous studies is that these research have focused on cybersecurity and anti-malware systems, which are designed to protect against external threats. However, this study demonstrates that LLMs are generally perceived as productivity-

enhancing tools rather than technologies to be avoided. This study respondents are developer dominated and they may view LLM risks negligible because they consider LLM outputs as manageable risks rather than severe threats, given their perceived response cost (mentioned below) of validating LLM output non-significant overhead cost. Our results suggest that in the software development context, developers do not perceive LLMs' risk as severe or likely enough to influence adoption intention. The productivity benefits and the popularity of LLMs could overshadow potential threats in the decision-making process.

Another explanation could be the lack of fear appeal in our setting. Unlike experiments that deliberately heighten fear appeals (Johnston & Warkentin, 2010), our survey did not promote the fear of negative consequences from flawed LLM output. Research on emergency warning apps (Fischer-Preßler et al., 2022) found that perceived severity failed to predict users' intentions when no strong fear trigger was present. Given the relative novelty of LLMs in professional software development environments, software professionals are not yet fully aware of the potential risks associated with LLM use. Awareness of specific threats such as hallucinations, biases, copyright infringement risks and security vulnerabilities are still developing within the community.

The hypothesis that response cost would negatively affect intention was unsupported. This indicates that perceived costs of human oversight did not significantly deter software professionals from intending to use LLMs. A plausible reason is that the sample in this study are mainly software developers, and human oversight is already an ingrained practice among developers, regardless of LLM use. In traditional software engineering workflows, critical code reviews, pair programming and quality assurance processes are common practice (Bacchelli &

Bird, 2013). Developers are accustomed to reviewing outputs as part of their work. Thus, the additional cost of reviewing LLM outputs may have been perceived as minimal or as expected, rather than an additional effort which would impact the adoption intention. In addition, the perceived benefits of using LLMs could outweigh concerns about the cost of oversight. It is plausible that developers prioritized productivity gains from LLM use over any concerns about oversight effort.

We hypothesized that higher efficacy beliefs would encourage adoption intention and reduce maladaptive coping. Specifically, greater response efficacy (belief in the effectiveness of human oversight) and self-efficacy (confidence in reviewing LLM outputs for errors, intervention points and legal compliance) could positively influence behavioral intention and negatively influence maladaptive coping. However, the findings did not support these relationships. One possible reason is that new risks often have low salience, and many individuals have not yet witnessed their consequences. Without first-hand experiences of threats, users may develop a false sense of security (Johnston & Warkentin, 2010). This might undermine the entire threat appraisal process (Johnston & Warkentin, 2010). In such cases, even if users agree that an oversight mechanism could be effective, the perceived threat remains too abstract to take any action. Essentially, strong efficacy beliefs are unlikely to motivate protective behavior if the person doubts the threat's relevance. This optimism bias in new technology contexts means efficacy constructs lack significance as long as people feel detached from the risks and do not act on coping appraisals (Johnston & Warkentin, 2010).

In addition, this study shows that LLM adoption decisions among software developers are likely benefit-driven rather than fear-driven. If an LLM tool greatly improves productivity, users may intend to use it regardless of how strenuous they

perceive the oversight. In such scenarios, the variance in behavioral intention is mostly captured by positive expectations which reduces the impact of response efficacy or self-efficacy.

In the literature, there is contextual variability in the predictive power of PMT constructs (Marikyan, 2023). This inconsistent pattern likely reflects how perceptions of threats and their consequences vary by technological context and application domain (Ifinedo, 2012).

In the LLM adoption context specifically, the absence of direct effects from PMT constructs on behavioral intention suggests that software professionals may process risk information differently than in other technological domains. The unprecedented productivity benefits and popularity of LLMs may overshadow potential threats in the decision-making process.

The relationship between behavioral intention and actual use emerged as one of the strongest in our model. This confirms that the intention to use LLMs predicts the actual adoption behavior among software professionals. This finding aligns with the existing literature which identified intention as a strong predictor for actual use .

6.2 Implications

6.2.1 Theoretical contribution

This research contributes to our understanding of how software development teams' approach LLM adoption in their work. By introducing a model based on PMT and UTAUT constructs, both risk perceptions and expected benefits have been applied to AI adoption decisions. Previous PMT-UTAUT models did not consider maladaptive coping construct which is included in this study and its effect was assessed. This

approach helps explain the considerations developers weigh when evaluating these AI tools.

Our application of PMT in the context of AI tools for professional software development environments extends the theory into a new domain. UTAUT constructs have been validated by this study in the context of AI.

6.2.2 Practical implications

For organizations and individuals, the study findings provide actionable insights. While LLM adoption is primarily benefit-driven, the lack of risk awareness among software professionals can lead to risky behaviors when interacting with LLMs such as integrating vulnerable code to projects or exploiting their sensitive business data to LLMs. Organizations might consider training programs for increasing the awareness of limitations and possible risks of LLM-generated outputs. Such initiatives could help foster a more informed usage, where LLMs are seen as valuable assistants, but not infallible sources.

Although perceived risks of using LLMs do not significantly affect adoption intention, they do have a significant effect on maladaptive coping behaviors. When risk perceptions of LLMs are more prevalent among software professionals but unaddressed, they may lead to avoidance, inaction, or deliberate neglect of those risks. Therefore, when organizations train software professionals on risks of LLMs, they should also incorporate risk mitigation strategies more clearly. This can ensure raising awareness of risks does not unintentionally discourage usage but instead promotes a responsible interaction with these tools.

6.3 Limitations and areas for further research

This study has several limitations. First, the use of non-probability convenience sampling limits the generalizability of findings to the broader software development community. As participants were selected based on accessibility rather than random sampling, the sample may not fully represent all demographic or professional groups.

In this study, the scope is expanded to include diverse roles within software development teams rather than focusing on specific occupational subgroups. Future research could benefit from examining adoption patterns within more narrowly defined professional roles to provide more granular insights into how different specialists approach LLM adoption. In addition, researchers may explore alternative coping responses to the risks discussed in this study. Identifying and evaluating different strategies beyond those examined here can enrich our understanding of how teams manage concerns related to LLM integration.

Future studies might consider tracking how professionals' views on LLM risks change over time as they gain real-world experience with these tools. A longitudinal study can be insightful. In addition, in-depth interviews with software professionals would likely provide more nuanced perspectives on what drives adoption decisions than our survey alone could capture. Therefore, we suggest including qualitative research methodology for the same study. Studies might also strengthen the model by including additional constructs that would influence technology acceptance. Furthermore, it would be valuable to replicate this study in the future within a different technological context.

APPENDIX A

CONSENT FORM

Consent:

Dear Participant,

This research was conducted within the scope of Boğaziçi University, Management Information Systems Master's program. It is carried out by Tuba Balta with the consultancy of Assoc. Prof. Dr. Nazım Taşkın. We aim to fill the gap in the literature with this Master Thesis on AI adoption in software development teams. We invite you to participate in this survey aimed at understanding your approach towards large language model outputs. If you agree to participate in this study, you will answer some questions related to your experience with LLM. The questionnaire consists of 47 questions. Filling the questionnaire is expected to take 15 minutes.

During the survey, no personal information will be collected. The results will be collected anonymously. Your participation in the study is voluntary. We will not charge or pay you for the participation. You have the right to withdraw from the project work and this survey at any time and in that case, you will not be negatively affected in any way. In addition, if you withdraw from the study at any point during the process, the data collected from your response will be destroyed, including the stored backups where the data is stored. The results of this research, which will analyze in depth the impact of flawed LLM output on the adoption of artificial intelligence, can be used by academics, potential artificial intelligence users and artificial intelligence suppliers for future studies and projects, and can be presented as a publication. Thank you for participating in our survey.

If you have any questions, please contact Tuba Balta (email: tuba.balta@std.bogazici.edu.tr) at any given time. You can also consult Boğaziçi University Social and Human Sciences Human Research Ethics Committee sbinarek@bogazici.edu.tr) regarding your rights in this study.

APPENDIX B

INDICATORS

Table 1 Survey constructs, questions and sources used in the research study.

Main Constructs	Indicators/items	References
Perceived Severity (PS)	PS1: I concern when projects are affected by using flawed LLM outputs PS2: I concern when the use of flawed LLM outputs hinders the project quality in the development process. PS3: I am scared that if I use flawed LLM outputs more frequently it will affect the project's reliability.	Fu, C. J., Silalahi, A. D. K., Huang, S. C., Phuong, D. T. T., Eunike, I. J., & Yu, Z. H. (2024). The (Un) Knowledgeable, the (Un) Skilled? Undertaking Chat-GPT Users' Benefit-Risk-Coping Paradox in Higher Education Focusing on an Integrated, UTAUT and PMT. International Journal of Human–Computer Interaction, 1-31.
Perceived Vulnerability (PV)	PV1. It is likely that I will use flawed LLM outputs in the projects PV2. It is possible that I will use flawed LLM outputs in the projects PV3. It is expected that I will use flawed LLM outputs in the projects PV4. If I use flawed LLM outputs in my projects, my career will be at risk	Rahi, S., Khan, M. M., & Alghizzawi, M. (2021). Factors influencing the adoption of telemedicine health services during COVID-19 pandemic crisis: an integrative research model. Enterprise Information Systems, 15(6), 769-793.
Response Efficacy (RE)	RE1. Implementing human oversight on Large Language Model (LLM) output effectively reduces the threat of flawed LLM outputs in projects. RE2. Using human oversight on LLM output will lower the probability of using flawed LLM outputs. RE3. Implementing human oversight on LLM output reduces the possibility of incorporating flawed LLM outputs in projects.	Chenoweth, T., Minch, R., & Gattiker, T. (2009, January). Application of protection motivation theory to adoption of protective technologies. In 2009 42nd Hawaii International Conference on System Sciences (pp. 1-10). IEEE.
Self-efficacy (SE)	SE1: I have the ability to successfully implement human oversight on Large Language Model outputs without assistance from others. SE2: I am confident that I have the ability to properly maintain and conduct human oversight of Large Language Model outputs. SE3: I feel I can implement human oversight for Large Language Model outputs without making mistakes. could contact someone when I got stuck	Chenoweth, T., Minch, R., & Gattiker, T. (2009, January). Application of protection motivation theory to adoption of protective technologies. In 2009 42nd Hawaii International Conference on System Sciences (pp. 1-10). IEEE.
Response Cost (RC)	RC1. There is too much overhead associated with implementing human oversight on LLM outputs in projects. RC2. It takes a considerable amount of time and effort to perform human oversight on LLM outputs. RC3. Human oversight on LLM output may cause distrustful outcome	Fu, C. J., Silalahi, A. D. K., Huang, S. C., Phuong, D. T. T., Eunike, I. J., & Yu, Z. H. (2024). The (Un) Knowledgeable, the (Un) Skilled? Undertaking Chat-GPT Users' Benefit-Risk-Coping Paradox in Higher Education Focusing on an Integrated, UTAUT and PMT. International Journal of Human–Computer Interaction, 1-31.
Performance Expectancy (PE)	PE1. I would find LLM useful in my job. PE2. Using LLM enables me to accomplish more tasks.	Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of

	PE3. Using LLM increases my productivity. PE4. If I use LLM on my projects, I will increase my chances of getting a raise.	information technology: Toward a unified view. MIS quarterly, 425-478.
Task Efficiency (TE)	TE1: I use LLM in my projects because it saves time completing my project tasks. TE2: I use LLM in my projects because it makes my project goal easier. TE3: I use LLM in my projects because it is useful for multitasking.	Fu, C. J., Silalahi, A. D. K., Huang, S. C., Phuong, D. T. T., Eunike, I. J., & Yu, Z. H. (2024). The (Un) Knowledgeable, the (Un) Skilled? Undertaking Chat-GPT Users' Benefit-Risk-Coping Paradox in Higher Education Focusing on an Integrated, UTAUT and PMT. International Journal of Human–Computer Interaction, 1-31.
Effort Expectancy (EE)	EE1. Learning how to use LLM is easy for me. EE2. My interaction with LLM is clear and understandable. EE3. I find LLM easy to use. EE4. It is easy for me to become skillful at using LLM.	Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. MIS quarterly, 425-478.
Maladaptive Coping (MC)	MC1. I try not to let the thought of flawed LLM outputs enter my mind MC2. I try not to think about the possibility of using flawed LLM outputs in my projects MC3. I try to ignore the possibility of using flawed LLM outputs	Chenoweth, T., Minch, R., & Gattiker, T. (2009, January). Application of protection motivation theory to adoption of protective technologies. In 2009 42nd Hawaii International Conference on System Sciences (pp. 1-10). IEEE.
Behavioral Intention (BI)	BI1: I intent to use LLMs in my projects in the future BI2: I plan to use LLMs in my projects in the future BI3:. I predict I would use LLMs in my projects in the future	Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. MIS quarterly, 425-478.
Actual Use (AU)	Please choose your usage frequency for LLM in your projects	Fu, C. J., Silalahi, A. D. K., Huang, S. C., Phuong, D. T. T., Eunike, I. J., & Yu, Z. H. (2024). The (Un) Knowledgeable, the (Un) Skilled? Undertaking Chat-GPT Users' Benefit-Risk-Coping Paradox in Higher Education Focusing on an Integrated, UTAUT and PMT. International Journal of Human–Computer Interaction, 1-31.

APPENDIX C

SURVEY QUESTIONS

Specify your gender: Female, Male, Prefer Not to say

Specify your age: • 22-25 • 25-29 • 30-35 • 36-41 • 42-50 • 51-60 • 60-65

Your Education Level:

- Primary/Middle School graduate
- High school graduate
- University student
- Graduated from a Universty
- Master's/PhD student
- Master's/PhD graduate

Please indicate how many years of total work experience you have:

- Less than 3 years ()
- 3 years or more, but less than 5 years ()
- 5 years or more, but less than 10 years ()
- 10 years or more, but less than 15 years ()
- More than 15 years ()

What is your current position in your company?

- Software Developer (Backend, Frontend, Full Stack, etc.)
- Data Science Specialist
- Product Owner
- Business Analyst
- DevOps Engineer
- Quality Assurance (QA) Engineer
- Project / Program Manager
- Technical Lead

Specify the industry your company operates in:

- Financial Services
- Technology, Media and Telecommunications
- Energy, Infrastructure and Natural Resources
- Retail and Consumer Products
- Automotive

- Industrial Production
- Construction and Engineering
- Health
- Transportation and Logistics
- Private Equity
- Government Policy
- Pharmaceuticals and Life Sciences
- Other

In which country is your company based?

Free text area

#	Indicators	Likert Scale
1	I would view the projects being affected by flawed Large Language Model (LLM) output as a serious problem.	[1] [2] [3] [4] [5] [6] [7]
2	Using flawed LLM outputs in the projects would be catastrophic.	[1] [2] [3] [4] [5] [6] [7]
3	Implementing flawed LLM outputs into the projects would have serious consequences.	[1] [2] [3] [4] [5] [6] [7]
4	I concern when projects are affected by using flawed LLM outputs	[1] [2] [3] [4] [5] [6] [7]
5	I concern when the use of flawed LLM outputs hinders the project quality in the development process.	[1] [2] [3] [4] [5] [6] [7]
6	I am scared that if I use flawed LLM outputs more frequently it will affect the project's reliability.	[1] [2] [3] [4] [5] [6] [7]
7	It is likely that I will use flawed LLM outputs in the projects	[1] [2] [3] [4] [5] [6] [7]
8	It is possible that I will use flawed LLM outputs in the projects	[1] [2] [3] [4] [5] [6] [7]
9	It is expected that I will use flawed LLM outputs in the projects	[1] [2] [3] [4] [5] [6] [7]
10	If I use flawed LLM outputs in my projects, my career will be at risk	[1] [2] [3] [4] [5] [6] [7]
11	Implementing human oversight on Large Language Model (LLM) output effectively reduces the threat of flawed LLM outputs in projects.	[1] [2] [3] [4] [5] [6] [7]

12	Using human oversight on LLM output will lower the probability of using flawed LLM outputs.	[1] [2] [3] [4] [5] [6] [7]
13	Implementing human oversight on LLM output reduces the possibility of incorporating flawed LLM outputs in projects.	[1] [2] [3] [4] [5] [6] [7]
14	Adopting Large Language Models will help me to develop cognitive ability	[1] [2] [3] [4] [5] [6] [7]
15	Implementing Large Language Models is an effective way to speed up completing my project	[1] [2] [3] [4] [5] [6] [7]
16	Enabling Large Language Models in projects is an effective way to gain new insights	[1] [2] [3] [4] [5] [6] [7]
17	I have the ability to successfully implement human oversight on LLM outputs without assistance from others.	[1] [2] [3] [4] [5] [6] [7]
18	I am confident that I have the ability to properly maintain and conduct human oversight of LLM outputs.	[1] [2] [3] [4] [5] [6] [7]
19	I feel I can implement human oversight for LLM outputs without making mistakes.	[1] [2] [3] [4] [5] [6] [7]
20	It would be easy for me to use Large Language Models in projects by myself	[1] [2] [3] [4] [5] [6] [7]
21	I could adopt Large Language Models in projects even though there is no one around to tell me what to do as I go.	[1] [2] [3] [4] [5] [6] [7]
22	I could adopt Large Language Models in projects if I could contact someone when I got stuck	[1] [2] [3] [4] [5] [6] [7]
23	There is too much overhead associated with implementing human oversight on Large Language Model (LLM) outputs in projects.	[1] [2] [3] [4] [5] [6] [7]
24	It takes a considerable amount of time and effort to perform human oversight on LLM outputs.	[1] [2] [3] [4] [5] [6] [7]
25	Human oversight on LLM output may cause distrustful outcome	[1] [2] [3] [4] [5] [6] [7]
26	I would find LLM useful in my job.	[1] [2] [3] [4] [5] [6] [7]
27	Using LLM enables me to accomplish more tasks.	[1] [2] [3] [4] [5] [6] [7]
28	Using LLM increases my productivity.	[1] [2] [3] [4] [5] [6] [7]

29	If I use LLM on my projects, I will increase my chances of getting a raise.	[1] [2] [3] [4] [5] [6] [7]
30	I use Large Language Model (LLM) in my projects because it saves time completing my project tasks.	[1] [2] [3] [4] [5] [6] [7]
31	I use LLM in my projects because it makes my project goal easier.	[1] [2] [3] [4] [5] [6] [7]
32	I use LLM in my projects because it is useful for multitasking.	[1] [2] [3] [4] [5] [6] [7]
33	Learning how to use LLM is easy for me.	[1] [2] [3] [4] [5] [6] [7]
34	My interaction with LLM is clear and understandable.	[1] [2] [3] [4] [5] [6] [7]
35	I find LLM easy to use.	[1] [2] [3] [4] [5] [6] [7]
36	It is easy for me to become skillful at using LLM.	[1] [2] [3] [4] [5] [6] [7]
37	I try not to let the thought of flawed Large Language Model (LLM) outputs enter my mind	[1] [2] [3] [4] [5] [6] [7]
38	I try not to think about the possibility of using flawed LLM outputs in my projects	[1] [2] [3] [4] [5] [6] [7]
39	I try to ignore the possibility of using flawed LLM outputs	[1] [2] [3] [4] [5] [6] [7]
40	I intend to use Large Language Models (LLMs) frequently in my projects.	[1] [2] [3] [4] [5] [6] [7]
41	I plan to integrate LLMs more extensively in my projects.	[1] [2] [3] [4] [5] [6] [7]
42	Whenever appropriate, I intend to utilize LLMs in my projects.	[1] [2] [3] [4] [5] [6] [7]
43	I expect to continue incorporating LLMs in my projects.	[1] [2] [3] [4] [5] [6] [7]
44	I intent to use LLMs in my projects in the future	[1] [2] [3] [4] [5] [6] [7]
45	I plan to use LLMs in my projects in the future	[1] [2] [3] [4] [5] [6] [7]
46	I predict I would use LLMs in my projects in the future	[1] [2] [3] [4] [5] [6] [7]
47	Please choose your usage frequency for LLM in your projects	Never

		Once a month Several times a month Once a week Several times a week Once a day Several times a day

APPENDIX D

COMBINED LOADINGS AND CROSS-LOADINGS

	PS	PV	RE	SE	RC	PE	EE	TE	MC	BI	AU
PS4	0,897	-0,01	0,033	-0,075	0,047	0,122	0,096	-0,137	-0,025	-0,144	0,04
PS5	0,893	-0,078	0,085	0,06	0,091	-0,035	-0,073	0,032	0,045	-0,061	0,125
PS6	0,839	0,094	-0,125	0,016	-0,148	-0,093	-0,025	0,113	-0,022	0,219	-0,176
PV1	-0,048	0,898	-0,071	-0,134	0,082	0,05	0,03	0,063	-0,064	-0,006	0,036
PV2	0,022	0,915	0,032	0,005	-0,027	0,123	-0,017	-0,212	0,007	0,026	-0,026
PV3	0,027	0,86	0,04	0,135	-0,056	-0,183	-0,013	0,16	0,059	-0,021	-0,01
RE1	-0,013	-0,03	0,851	0,179	0,026	0,209	-0,082	-0,086	-0,026	-0,059	0,048
RE2	0,111	-0,012	0,835	-0,201	0,012	0,003	0,114	0,039	0,038	-0,045	-0,037
RE3	-0,089	0,039	0,915	0,017	-0,035	-0,197	-0,027	0,044	-0,011	0,096	-0,011
SE1	0,051	0,053	-0,023	0,896	-0,018	0,041	-0,22	-0,149	-0,039	0,08	0,14
SE2	0,019	0,015	0,082	0,926	0,027	0,126	-0,002	-0,115	0,003	-0,029	0,104
SE3	-0,074	-0,072	-0,065	0,854	-0,011	-0,179	0,233	0,281	0,038	-0,052	-0,26
RC1	-0,1	0,003	0,136	-0,119	0,851	0,086	0,146	-0,07	-0,124	-0,129	0,068
RC2	0,079	-0,084	0,01	0,076	0,893	0,152	-0,065	-0,081	0,064	-0,06	0,009
RC3	0,022	0,11	-0,187	0,051	0,663	-0,316	-0,099	0,198	0,072	0,247	-0,098
PE1	0,026	0,053	0,004	0,002	-0,061	0,943	0,012	-0,129	-0,013	0,151	-0,021
PE2	-0,026	-0,053	-0,004	-0,002	0,061	0,943	-0,012	0,129	0,013	-0,151	0,021
EE1	0,038	-0,057	0,007	0,106	0,019	0,357	0,889	-0,132	0,02	-0,043	0,127
EE2	0,097	-0,049	0,002	-0,05	-0,028	-0,088	0,93	0,075	0,035	0,032	-0,04
EE3	-0,018	0,062	-0,003	-0,063	-0,012	-0,159	0,934	0,002	-0,061	0,059	-0,06
EE4	-0,119	0,042	-0,007	0,013	0,022	-0,096	0,904	0,051	0,008	-0,051	-0,021

TE1	-0,01	-0,01	0,043	0,017	0,069	0,238	-0,019	0,909	-0,018	0,036	0,102
TE2	0,007	0,049	-0,012	-0,108	-0,013	0,046	0,017	0,937	-0,021	0,059	0,009
TE3	0,003	-0,041	-0,031	0,095	-0,056	-0,288	0,001	0,902	0,04	-0,098	-0,112
MC1	0,04	-0,053	0,024	0,075	0,028	-0,4	0,03	0,331	0,826	-0,035	0,113
MC2	0,038	0,077	-0,017	0,006	-0,061	0,138	-0,049	-0,101	0,901	0,084	-0,058
MC3	-0,076	-0,029	-0,005	-0,076	0,036	0,231	0,021	-0,205	0,89	-0,053	-0,046
BI5	-0,006	0,039	-0,014	0,014	-0,023	-0,081	0,073	0,144	-0,017	0,93	-0,113
BI6	0,021	0,001	-0,015	-0,023	-0,014	0,056	0,021	0,008	0,024	0,955	0,05
BI7	-0,016	-0,041	0,03	0,01	0,038	0,024	-0,096	-0,153	-0,007	0,922	0,063
AU	0	0	0	0	0	0	0	0	0	0	1.000

Combined loadings and cross-loadings P Values

	Type	SE	P value
PS4	Reflect	0,067	<0,001
PS5	Reflect	0,067	<0,001
PS6	Reflect	0,068	<0,001
PV1	Reflect	0,067	<0,001
PV2	Reflect	0,066	<0,001
PV3	Reflect	0,067	<0,001
RE1	Reflect	0,067	<0,001
RE2	Reflect	0,068	<0,001
RE3	Reflect	0,066	<0,001
SE1	Reflect	0,067	<0,001
SE2	Reflect	0,066	<0,001

SE3	Reflect	0,067	<0,001
RC1	Reflect	0,067	<0,001
RC2	Reflect	0,067	<0,001
RC3	Reflect	0,07	<0,001
PE1	Reflect	0,066	<0,001
PE2	Reflect	0,066	<0,001
EE1	Reflect	0,067	<0,001
EE2	Reflect	0,066	<0,001
EE3	Reflect	0,066	<0,001
EE4	Reflect	0,067	<0,001
TE1	Reflect	0,067	<0,001
TE2	Reflect	0,066	<0,001
TE3	Reflect	0,067	<0,001
MC1	Reflect	0,068	<0,001
MC2	Reflect	0,067	<0,001
MC3	Reflect	0,067	<0,001
BI5	Reflect	0,066	<0,001
BI6	Reflect	0,066	<0,001
BI7	Reflect	0,066	<0,001
ACTUAL_USE	Reflect	0,065	<0,001

APPENDIX E

ETHICS COMMITTEE APPROVAL

T.C.
BOĞAZİÇİ ÜNİVERSİTESİ REKTÖRLÜĞÜ
Sosyal ve Beşeri Bilimler İnsan Araştırmaları Etik Kurulu (Sbinarek)

2025 AİLE YILI

Sayı : E-84391427-050.04-222263 25.02.2025
Konu : 2024-91T Kayıt Numaralı Başvurunuz Hakkında

Sayın Doç. Dr. Nazım TAŞKIN

Tez danışmanlığını yürüttüğünüz öğrenciniz Tuba Balta'nın "AI adoption in software development teams" başlıklı projesi ile Boğaziçi Üniversitesi Sosyal ve Beşeri Bilimler İnsan Araştırmaları Etik Kurulu (SBİNAREK)'e yaptığınız 2024-91T kayıt numaralı başvuru 17.02.2025 tarih ve 2025/02 sayılı kurul toplantısında incelenmiş ve projeye etik onay verilmesi uygun bulunmuştur.

Bu karar tüm üyelerin toplantıya on-line olarak katılımıyla ve oybirliği ile alınmıştır. Onay mektubu tüm üyeler adına Komisyon Başkanı tarafından e-imzalanmıştır. Bilgilerinizi rica ederiz.

Saygılarımızla,

Dr. Öğr. Üyesi Işıl ERDUYAN
Kurul Başkanı

Bu belge, güvenli elektronik imza ile imzalanmıştır.
Belge Doğrulama Kodu : 0B33-E76M-0HKE Belge Doğrulama Adresi : https://www.turkiye.gov.tr/bogazici-universitesi-ebys

Adres: Boğaziçi Üniversitesi 34342 Bebek/ İstanbul
Telefon No: Faks No:
e-Posta: İnternet Adresi: www.bogazici.edu.tr
Kep Adresi: bogaziciuniversitesi@hs01.kep.tr

Bilgi İçin: Nurşen MUNAR
Mühendis
Telefon No: 0 212 359-6749

REFERENCES

Adiguzel, T., Kaya, M. H., & Cansu, F. K. (2023). Revolutionizing education with AI: Exploring the transformative potential of ChatGPT. *Contemporary Educational Technology*, *15*(3), ep429. https://doi.org/10.30935/cedtech/13152

AI RMF - AIRC. (n.d.). NIST AI Resource Center. Retrieved March 30, 2025, from https://airc.nist.gov/airmf-resources/airmf/

Almeida, A., Xavier, L., & Valente, M. T. (2024). Automatic library migration using large language models: First results. *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 427–433. https://doi.org/10.1145/3674805.3690746

Alneyadi, M. A. H., Rashed, M., & Normalini, M. K. (2023). Factors influencing user's intention to adopt AI-based cybersecurity systems in the UAE. *Interdisciplinary Journal of Information, Knowledge, and Management*, *18*, 459–486.

Alsaqqa, S., Sawalha, S., & Abdel-Nabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies (iJIM)*, *14*(11), Article 11. https://doi.org/10.3991/ijim.v14i11.13269

Al-Sharafi, M. A., Al-Emran, M., Arpaci, I., Iahad, N. A., AlQudah, A. A., Iranmanesh, M., & Al-Qaysi, N. (2023). Generation Z use of artificial intelligence products and its impact on environmental sustainability: A cross-cultural comparison. *Computers in Human Behavior*, *143*, 107708. https://doi.org/10.1016/j.chb.2023.107708

Alter, A., & Harris, E. A. (2023, September 20). Franzen, Grisham and other prominent authors sue OpenAI. *The New York Times*. https://www.nytimes.com/2023/09/20/books/authors-openai-lawsuit-chatgpt-copyright.html

Amora, J. T. (2021). *Convergent validity assessment in PLS-SEM: A loadings-driven approach*.

Anderson, C. L., & Agarwal, R. (2010). Practicing safe computing: A multimethod empirical examination of home computer user security behavioral intentions. *MIS Quarterly*, *34*(3), 613–643. https://doi.org/10.2307/25750694

Armstrong, J. S., & Overton, T. S. (1977). Estimating nonresponse bias in mail surveys. *Journal of Marketing Research*, *14*(3), 396–402. https://doi.org/10.1177/002224377701400320

Article 14: Human Oversight | EU Artificial Intelligence Act. (n.d.). Retrieved January 2, 2025, from https://artificialintelligenceact.eu/article/14/

Madiega, T. (2024). *Artificial intelligence act*. (PE 698.792). European Parliamentary Research Service. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R1689

Aschenbrenner, D., & Colloseus, C. (2023). Human in command in manufacturing. In E. Alfnes, A. Romsdal, J. O. Strandhagen, G. von Cieminski, & D. Romero (Eds.), *Advances in Production Management Systems. Production Management Systems for Responsible Manufacturing, Service, and Logistics Futures* (pp. 559–572). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-43662-8_40

Business Analysis Body of Knowledge (BABOK). (n.d.). Retrieved March 2, 2025, https://moodle.znu.edu.ua/pluginfile.php/196199/mod_resource/content/3/BABOK.pdf

Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. *2013 35th International Conference on Software Engineering (ICSE)*, 712–721. https://doi.org/10.1109/ICSE.2013.6606617

Bajohr, H. (2022). Algorithmic empathy: Toward a critique of aesthetic AI. *Configurations*, *30*(2), 203–231.

Balaji, S., & Murugaiyan, M. S. (2012). *Wateerfall vs V-model vs Agile: A comparative study on SDLC*. International Journal of Information Technology and Business Management, 2(1), 1–7. http://www.jitbm.com/Volume2/1.pdf.

Bandura, A. (1978). Self-efficacy: Toward a unifying theory of behavioral change. *Advances in Behaviour Research and Therapy*, *1*(4), 139–161. https://doi.org/10.1016/0146-6402(78)90002-4

Bekbolatova, M., Mayer, J., Ong, C. W., & Toma, M. (2024). Transformative potential of AI in healthcare: Definitions, applications, and navigating the ethical landscape and public perspectives. *Healthcare*, *12*(2), 125. https://doi.org/10.3390/healthcare12020125

Belzner, L., Gabor, T., & Wirsing, M. (2024). Large language model assisted software engineering: Prospects, challenges, and a case study. In B. Steffen (Ed.), *Bridging the Gap Between AI and Reality* (Vol. 14380, pp. 355–374). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-46002-9_23

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). *On the dangers of stochastic parrots: Can language models be too big?*. *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 610–623. https://doi.org/10.1145/3442188.3445922

Bengio, Y., Ducharme, R., & Vincent, P. (2000). A neural probabilistic language model. *Advances in Neural Information Processing Systems*, *13*. https://proceedings.neurips.cc/paper_files/paper/2000/hash/728f206c2a01bf572b5940d7d9a8fa4c-Abstract.html

Blut, M., Durham University Business School, UK, Chong, A. Y. L., Nottingham University Business School–Ningbo, China, Tsigna, Z., Nottingham University Business School–Ningbo, China, Venkatesh, V., & Pamplin College of Business, Virginia Tech, USA. (2022). Meta-analysis of the unified theory of acceptance and use of technology (UTAUT): Challenging its validity and charting a research agenda in the red ocean. *Journal of the Association for Information Systems*, *23*(1), 13–95. https://doi.org/10.17705/1jais.00719

Borji, A. (2023). *A Categorical Archive of ChatGPT Failures* (No. arXiv:2302.03494). arXiv. https://doi.org/10.48550/arXiv.2302.03494

Borkar, J. (2023). *What can we learn from Data Leakage and Unlearning for Law?* (No. arXiv:2307.10476). arXiv. https://doi.org/10.48550/arXiv.2307.10476

Boss, S. R., Galletta, D. F., Lowry, P. B., Moody, G. D., & Polak, P. (2015). *What do systems users have to fear? Using fear appeals to engender threats and fear that motivate protective security behaviors*. *MIS Quarterly*, *39*(4), 837–864.

Brislin, R. W. (1970). Back-translation for cross-cultural research. *Journal of Cross-Cultural Psychology*, *1*(3), 185–216. https://doi.org/10.1177/135910457000100301

Brown, P. F., Della Pietra, V. J., deSouza, P. V., Lai, J. C., & Mercer, R. L. (1992). Class-Based *n*-gram Models of Natural Language. *Computational Linguistics*, *18*(4), 467–480.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., … Amodei, D. (2020). Language models are few-shot earners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

California Consumer Privacy Act of 2018. (n.d.).

Camilleri, M. A. (2024). *Factors affecting performance expectancy and intentions to use ChatGPT: Using SmartPLS to advance an information technology acceptance framework—ScienceDirect*. https://www.sciencedirect.com/science/article/pii/S004016252400043X

Cao, G., Duan, Y., Edwards, J. S., & Dwivedi, Y. K. (2021). Understanding managers' attitudes and behavioral intentions towards using artificial intelligence for organizational decision-making. *Technovation*, *106*, 102312. https://doi.org/10.1016/j.technovation.2021.102312

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A., & Raffel, C. (2021). *Extracting training data from large language models*. In *Proceedings of the 30th USENIX Security Symposium* (pp. 2633–2648). USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting

ChatGPT and large language models: What's the risk? (n.d.). Retrieved December 30, 2024, from https://www.ncsc.gov.uk/blog-post/chatgpt-and-large-language-models-whats-the-risk

Chen, L., Xu, M., & Li, X. (2024). Promethean Anxiety from AI Adoption: Employees' Need for Self-Actualization and Turnover Intention. *Academy of Management Proceedings*, *2024*(1), 15185. https://doi.org/10.5465/AMPROC.2024.218bp

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021a). *Evaluating Large Language Models Trained on Code* (No. arXiv:2107.03374). arXiv. https://doi.org/10.48550/arXiv.2107.03374

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021b). *Evaluating Large Language Models Trained on Code* (No. arXiv:2107.03374). arXiv. https://doi.org/10.48550/arXiv.2107.03374

Chen, Y., Hu, Z., Zhi, C., Han, J., Deng, S., & Yin, J. (2024). *ChatUniTest: A Framework for LLM-Based Test Generation* (No. arXiv:2305.04764). arXiv. https://doi.org/10.48550/arXiv.2305.04764

Chen, Y., Luo, X. (Robert), & Li, H. (2022). Beyond adaptive security coping behaviors: Theory and empirical evidence. *Information & Management*, *59*(2), 103575. https://doi.org/10.1016/j.im.2021.103575

Chen, Y., & Zahedi, F. M. (2016). Individuals' Internet Security Perceptions and Behaviors: Polycontextual Contrasts Between the United States and China. *MIS Quarterly*, *40*(1), 205–222.

Chenoweth, T., Gattiker, T., & Corral, K. (2019). Adaptive and Maladaptive Coping with an It Threat. *Information Systems Management*, *36*(1), 24–39. https://doi.org/10.1080/10580530.2018.1553647

Chenoweth, T., Minch, R., & Gattiker, T. (2009). Application of Protection Motivation Theory to Adoption of Protective Technologies. *2009 42nd Hawaii International Conference on System Sciences*, 1–10. https://doi.org/10.1109/HICSS.2009.74

Chin, W. W., Marcolin, B. L., & Newsted, P. R. (2003). A partial least squares latent variable modeling approach for measuring interaction effects: Results from a Monte Carlo simulation study and an electronic-mail emotion/adoption study. *Information Systems Research*, *14*(2), 189–217. https://doi.org/10.1287/isre.14.2.189.16018

Chin, W. W., & Newsted, P. R. (1999). Structural equation modeling analysis with small samples using partial least squares. *Statistical strategies for small sample research* (pp. 307–341). Sage Publications.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., … Fiedel, N. (2023). PaLM: Scaling language modeling with Pathways. *Journal of Machine Learning Research*, *24*(240), 1–113.

Cosler, M., Hahn, C., Mendoza, D., Schmitt, F., & Trippel, C. (2023). nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. In C. Enea & A. Lal (Eds.), *Computer aided verification* (pp. 383–396). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-37703-7_18

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches (5th ed.).* SAGE Publications.

Crossler, R., & Bélanger, F. (2014). An extended perspective on individual security behaviors: Protection Motivation Theory and a unified security practices (USP) instrument. *SIGMIS Database*, *45*(4), 51–71. https://doi.org/10.1145/2691517.2691521

Crossler, R. E. (2010). Protection Motivation Theory: Understanding determinants to backing up personal data. In *2010 43rd Hawaii International Conference on System Sciences* (pp. 1–10). https://doi.org/10.1109/HICSS.2010.311

Dastin, J. (2022). Amazon scraps secret AI recruiting tool that showed bias against women. In Ethics of data and analytics (pp. 296-299). Auerbach Publications.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of deep bidirectional transformers for language understanding.* https://arxiv.org/abs/1810.04805

Diggs, C., Doyle, M., Madan, A., Scott, S., Escamilla, E., Zimmer, J., Nekoo, N., Ursino, P., Bartholf, M., Robin, Z., Patel, A., Glasz, C., Macke, W., Kirk, P., Phillips, J., Sridharan, A., Wendt, D., Rosen, S., Naik, N., … Thaker, S. (2024). Leveraging LLMs for legacy code modernization: Challenges and opportunities for LLM-generated documentation (No. arXiv:2411.14971). *arXiv*. https://doi.org/10.48550/arXiv.2411.14971

Dou, S., Shan, J., Jia, H., Deng, W., Xi, Z., He, W., Wu, Y., Gui, T., Liu, Y., & Huang, X. (2023). *Towards understanding the capability of large language models on code clone detection: A survey (No. arXiv:2308.01191). arXiv.* https://doi.org/10.48550/arXiv.2308.01191

Dressel, J., & Farid, H. (2018). The accuracy, fairness, and limits of predicting recidivism. *Science Advances*, *4*(1), eaao5580. https://doi.org/10.1126/sciadv.aao5580

Dvivedi, S. S., Vijay, V., Pujari, S. L. R., Lodh, S., & Kumar, D. (2024). A comparative analysis of large language models for code documentation generation. In *Proceedings of the 1st ACM International Conference on AI-Powered Software* (pp. 65–73). https://doi.org/10.1145/3664646.3664765

Dwivedi, Y. K., Rana, N. P., Jeyaraj, A., Clement, M., & Williams, M. D. (2019). Re-examining the unified theory of acceptance and use of technology (UTAUT): Towards a revised theoretical model. *Information Systems Frontiers, 21*(3), 719–734. https://doi.org/10.1007/s10796-017-9774-y

Fan, A., Gokkaya, B., Harman, M., Lyubarskiy, M., Sengupta, S., Yoo, S., & Zhang, J. M. (2023). Large language models for software engineering: Survey and open problems. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)* (pp. 31–53). https://doi.org/10.1109/ICSE-FoSE59343.2023.00008

Feretzakis, G., Vagena, E., Kalodanis, K., Peristera, P., Kalles, D., & Anastasiou, A. (2025). GDPR and large language models: Technical and legal obstacles. *Future Internet, 17*(4), Article 4. https://doi.org/10.3390/fi17040151

Fischer-Preßler, D., Bonaretti, D., & Fischbach, K. (2022). A protection-motivation perspective to explain intention to use and continue to use mobile warning systems. *Business & Information Systems Engineering, 64*(2), 167–182. https://doi.org/10.1007/s12599-021-00704-0

Fogel, D. B. (2022). Defining artificial intelligence. In *Machine learning and the city* (pp. 91–120). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781119815075.ch7

Fu, C.-J., Silalahi, A. D. K., Huang, S.-C., Phuong, D. T. T., Eunike, I. J., & Yu, Z.-H. (2024). The (un)knowledgeable, the (un)skilled? Undertaking Chat-GPT users' benefit-risk-coping paradox in higher education focusing on an integrated, UTAUT and PMT. *International Journal of Human–Computer Interaction*, *0*(0), 1–31. https://doi.org/10.1080/10447318.2024.2365028

Fu, Y., Liang, P., Tahir, A., Li, Z., Shahin, M., Yu, J., & Chen, J. (2025). Security weaknesses of Copilot-generated code in GitHub projects: An empirical study. *ACM Transactions on Software Engineering and Methodology.* https://doi.org/10.1145/3716848

Fuster, A., Goldsmith-Pinkham, P., Ramadorai, T., & Walther, A. (2022). Predictably unequal? The effects of machine learning on credit markets. *The Journal of Finance*, *77*(1), 5–47. https://doi.org/10.1111/jofi.13090

Garcia, A. C. B., Garcia, M. G. P., & Rigobon, R. (2024). Algorithmic discrimination in the credit domain: What do we know about it? *AI & Society, 39*(4), 2059–2098. https://doi.org/10.1007/s00146-023-01676-3

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, *2*(11), 665–673. https://doi.org/10.1038/s42256-020-00257-z

General Data Protection Regulation (GDPR) – Legal Text. (n.d.). General Data Protection Regulation (GDPR). Retrieved December 30, 2024, from https://gdpr-info.eu/

Goanta, C., Aletras, N., Chalkidis, I., Ranchordas, S., & Spanakis, G. (2023). *Regulation and NLP (RegNLP): Taming large language models (No. arXiv:2310.05553). arXiv.* https://doi.org/10.48550/arXiv.2310.05553

Gokarna, M., & Singh, R. (2021). DevOps: A historical review and future works. In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)* (pp. 366–371). https://doi.org/10.1109/ICCCIS51004.2021.9397235

Gonzales, J. T. (2023). Implications of AI innovation on economic growth: A panel data study. *Journal of Economic Structures*, *12*(1), 13. https://doi.org/10.1186/s40008-023-00307-w

Gopal, A., Bostrom ,Robert P., & and Chin, W. W. (1992). Applying adaptive structuration theory to investigate the process of group support systems use. *Journal of Management Information Systems, 9*(3), 45–69. *Journal of Management Information Systems*, *9*(3), 45–69. https://doi.org/10.1080/07421222.1992.11517967

Goswami, V., Malviya, V., & Sharma, P. (2020). Detecting spam emails/SMS using Naive Bayes, support vector machine and random forest. In A. P. Pandian, R. Palanisamy, & K. Ntalianis (Eds.), *Proceedings of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2019)* (pp. 306–313). Springer International Publishing. https://doi.org/10.1007/978-3-030-43192-1_35

Gruetzemacher, R., & Whittlestone, J. (2019). *Defining and unpacking transformative AI.* https://www.researchgate.net/publication/337702892_Defining_and_Unpacking_Transformative_AI

Grynbaum, M. M., & Mac, R. (2023, December 27). The Times sues OpenAI and Microsoft over A.I. use of copyrighted work. *The New York Times*. https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html

Hacker, P., Engel, A., Hammer, S., & Mittelstadt, B. (2025). *Introduction to the foundations and regulation of generative AI (SSRN Scholarly Paper No. 5137750). Social Science Research Network.* https://doi.org/10.2139/ssrn.5137750

Hair, J. F., Hult, G. T. M., Ringle, C. M., & Sarstedt, M. (2022). *A primer on partial least squares structural equation modeling (PLS-SEM)* (3rd Edition, Vol. 38). http://www.tandfonline.com/doi/abs/10.1080/1743727X.2015.1005806

Hair, J. F., Ringle, C. M., & Sarstedt, M. (2011). PLS-SEM: Indeed a Silver Bullet. *Journal of Marketing Theory and Practice*, *19*(2), 139–152. https://doi.org/10.2753/MTP1069-6679190202

Hair, J. F., Ringle ,Christian M., & and Sarstedt, M. (2011). PLS-SEM: Indeed a silver bullet. *Journal of Marketing Theory and Practice, 19*(2), 139–152. https://doi.org/10.2753/MTP1069-6679190202

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In T. Hastie, R. Tibshirani, & J. Friedman, *The elements of statistical learning* (pp. 485–585). Springer New York. https://doi.org/10.1007/978-0-387-84858-7_14

Henseler, J., Ringle, C. M., & Sarstedt, M. (2015). A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the Academy of Marketing Science*, *43*(1), 115–135. https://doi.org/10.1007/s11747-014-0403-8

Henseler, J., Ringle, C. M., & Sinkovics, R. R. (2009). The use of partial least squares path modeling in international marketing. In R. R. Sinkovics & P. N. Ghauri (Eds.), *Advances in International Marketing* (Vol. 20, pp. 277–319). Emerald Group Publishing Limited. https://doi.org/10.1108/S1474-7979(2009)0000020014

Hern, A. (2022, May 25). TechScape: Clearview AI was fined £7.5m for brazenly harvesting your data – does it care? *The Guardian*. https://www.theguardian.com/technology/2022/may/25/techscape-clearview-ai-facial-recognition-fine

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hossain, S. B., Jiang, N., Zhou, Q., Li, X., Chiang, W.-H., Lyu, Y., Nguyen, H., & Tripp, O. (2024). A deep dive into large language models for automated bug localization and repair. *Proceedings of the ACM on Software Engineering, 1*(FSE), 66:1471–66:1493. https://doi.org/10.1145/3660773

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology, 33*(8), 1–79. https://doi.org/10.1145/3695988

Howard, M. C. (2016). A review of exploratory factor analysis decisions and overview of current practices: What we are doing and how can we improve? *International Journal of Human–Computer Interaction, 32*(1), 51–62. https://doi.org/10.1080/10447318.2015.1087664

Hsu, W.-L., & Silalahi, A. D. K. (2024). Exploring the paradoxical use of ChatGPT in education: Analyzing benefits, risks, and coping strategies through integrated UTAUT and PMT theories using a hybrid approach of SEM and fsQCA. *Computers and Education: Artificial Intelligence*, *7*, 100329. https://doi.org/10.1016/j.caeai.2024.100329

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2025). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems, 43*(2), 1–55. https://doi.org/10.1145/3703155

IEEE. (1990). *IEEE Std 610.12-1990: IEEE standard glossary of software engineering terminology* (pp. 1–84). *IEEE*. https://doi.org/10.1109/IEEESTD.1990.101064

Ifinedo, P. (2012). Understanding information systems security policy compliance: An integration of the theory of planned behavior and the protection motivation theory. *Computers & Security*, *31*(1), 83–95. https://doi.org/10.1016/j.cose.2011.10.007

Illman, E., & Temple, P. (2019). California Consumer Privacy Act: What companies need to know. *The Business Lawyer*, *75*(1), 1637–1646.

Jelinek, F. (1998). *Statistical methods for speech recognition*. MIT Press.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys, 55*(12), 1–38. https://doi.org/10.1145/3571730

Jiang, J., Wang, F., Shen, J., Kim, S., & Kim, S. (2024). *A survey on large language models for code generation (No. arXiv:2406.00515). arXiv.* https://doi.org/10.48550/arXiv.2406.00515

Johnston, A. C., & Warkentin, M. (2010). Fear appeals and information security behaviors: An empirical study. *MIS Quarterly*, *34*(3), 549–566. https://doi.org/10.2307/25750691

Kaber, D. B., & Endsley, M. R. (2004). The effects of level of automation and adaptive automation on human performance, situation awareness and

workload in a dynamic control task. *Theoretical Issues in Ergonomics Science*, *5*(2), 113–153. https://doi.org/10.1080/1463922021000054335

Karamolegkou, A., Li, J., Zhou, L., & Søgaard, A. (2023). *Copyright violations and large language models (No. arXiv:2310.13771). arXiv.* https://doi.org/10.48550/arXiv.2310.13771

Khalilzadeh, J., Ozturk, A. B., & Bilgihan, A. (2017). Security-related factors in extended UTAUT model for NFC based mobile payment in the restaurant industry. *Computers in Human Behavior*, *70*, 460–474. https://doi.org/10.1016/j.chb.2017.01.001

Khan, J. Y., & Uddin, G. (2023). Automatic code documentation generation using GPT-3. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 1–6. https://doi.org/10.1145/3551349.3559548

Khan, W. Z., Kibriya, H., Siddiqa, A., & Khurram khan, M. (2024). *Privacy issues in large language models: A survey (SSRN Scholarly Paper No. 4871294). Social Science Research Network.* https://doi.org/10.2139/ssrn.4871294

Kock, N. (2015). Common method bias in PLS-SEM: A full collinearity assessment approach. *International Journal of E-Collaboration, 11*(4), 1–10. https://doi.org/10.4018/ijec.2015100101

Kock, N. (2018). Should bootstrapping be used in PLS-SEM? Toward stable p-value calculation methods. *Journal of Applied Structural Equation Modeling, 2*(1), 1–12. https://doi.org/10.47263/JASEM.2(1)02

Kock, N. (2021). *Harman's single factor test in PLS-SEM: Checking for common method bias. Data Analysis Perspectives Journal, 2(2), 1–6.* https://www.scriptwarp.com

Kock, N. (2024). *WarpPLS User Manual: Version 8.0*. ScriptWarp Systems.

Kock, N., & Hadaya, P. (2018). Minimum sample size estimation in PLS-SEM: The inverse square root and gamma-exponential methods. *Information Systems Journal*, *28*(1), 227–261. https://doi.org/10.1111/isj.12131

Kock, N., & Lynn, G. (2012). Lateral collinearity and misleading results in variance-based SEM: An illustration and recommendations. *Journal of the Association for Information Systems, 13*(7). https://doi.org/10.17705/1jais.00302

Krishna, M., Gaur, B., Verma, A., & Jalote, P. (2024). Using LLMs in software requirements specifications: An empirical evaluation. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, 475–483. https://doi.org/10.1109/RE59067.2024.00056

Kuhail, M. A., Mathew, S. S., Khalil, A., Berengueres, J., & Shah, S. J. H. (2024). "Will I be replaced?" Assessing ChatGPT's effect on software development and programmer perceptions of AI tools. *Science of Computer Programming*, *235*, 103111. https://doi.org/10.1016/j.scico.2024.103111

Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for t-tests and ANOVAs. *Frontiers in Psychology*, *4*. https://doi.org/10.3389/fpsyg.2013.00863

Langer, M., Baum, K., & Schlicker, N. (2024). Effective human oversight of AI-based systems: A signal detection perspective on the detection of inaccurate and unfair outputs. *Minds and Machines*, *35*(1), 1. https://doi.org/10.1007/s11023-024-09701-0

Lazarus, R. S., & Folkman, S. (1984). *Stress, Appraisal, and Coping*. Springer.

Lee, C., Xia, C. S., Yang, L., Huang, J., Zhu, Z., Zhang, L., & Lyu, M. R. (2024). *A unified debugging approach via LLM-based multi-agent synergy (No. arXiv:2404.17153). arXiv.* https://doi.org/10.48550/arXiv.2404.17153

Lee, Y., Jeong, S., & Kim, J. (2024). Improving LLM classification of logical errors by integrating error relationship into prompts. In A. Sifaleras & F. Lin (Eds.), *Generative intelligence and intelligent tutoring systems* (pp. 91–103). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-63028-6_8

Li, Y., Keung, J., Ma, X., Chong, C. Y., Zhang, J., & Liao, Y. (2024). LLM-based class diagram derivation from user stories with chain-of-thought promptings. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 45–50). https://doi.org/10.1109/COMPSAC61105.2024.00017

Li, Z. (2023). *The dark side of ChatGPT: Legal and ethical challenges from stochastic parrots and hallucination (No. arXiv:2304.14347). arXiv.* https://doi.org/10.48550/arXiv.2304.14347

Liang, H., & Xue, Y. (2009). Avoidance of information technology threats: A theoretical perspective. *MIS Quarterly*, *33*(1), 71–90. https://doi.org/10.2307/20650279

Liang, H., Xue, Y., Pinsonneault, A., & Wu, Y. "Andy." (2019). What users do besides problem-focused coping when facing IT security threats: An emotion-focused coping perspective. *MIS Quarterly*, *43*(2), 373-A18.

Liu, F., Liu, Y., Shi, L., Huang, H., Wang, R., Yang, Z., Zhang, L., Li, Z., & Ma, Y. (2024). *Exploring and evaluating hallucinations in LLM-powered code generation (No. arXiv:2404.00971). arXiv.* https://doi.org/10.48550/arXiv.2404.00971

Liu, X., Sun, T., Xu, T., Wu, F., Wang, C., Wang, X., & Gao, J. (2024). *SHIELD: Evaluation and defense strategies for copyright compliance in LLM text generation (No. arXiv:2406.12975). arXiv.* https://doi.org/10.48550/arXiv.2406.12975

Liu, Y., He, H., Han, T., Zhang, X., Liu, M., Tian, J., Zhang, Y., Wang, J., Gao, X., Zhong, T., Pan, Y., Xu, S., Wu, Z., Liu, Z., Zhang, X., Zhang, S., Hu, X., Zhang, T., Qiang, N., … Ge, B. (2024). *Understanding LLMs: A comprehensive overview from training to inference (No. arXiv:2401.02038). arXiv.* https://doi.org/10.48550/arXiv.2401.02038

Ma, W., Song, Y., Xue, M., Wen, S., & Xiang, Y. (2024). The "code" of ethics: A holistic audit of AI code generators. *IEEE Transactions on Dependable and Secure Computing, 21*(5), 4997–5013. IEEE Transactions on Dependable and Secure Computing. https://doi.org/10.1109/TDSC.2024.3367737

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., & Clark, P. (2023). Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems, 36*, 46534–46594.

Maldonado, E. da S., Shihab, E., & Tsantalis, N. (2017). Using natural language processing to automatically detect self-admitted technical debt. *IEEE*

Transactions on Software Engineering, *43*(11), 1044–1062. https://doi.org/10.1109/TSE.2017.2654244

Marikyan, D., & Papagiannidis, S. (2023). *Protection Motivation Theory: A review*. In S. Papagiannidis (Ed.), *TheoryHub Book* (pp. 78–93). TheoryHub. https://open.ncl.ac.uk/theory-library/TheoryHubBook.pdf.

Marques, N., Silva, R. R., & Bernardino, J. (2024). Using ChatGPT in software requirements engineering: A comprehensive review. *Future Internet*, *16*(6), Article 6. https://doi.org/10.3390/fi16060180

Matulionyte, R., & Lee, J.-A. (2022a). Copyright in AI-generated works: Lessons from recent developments in patent law. *SCRIPTed: A Journal of Law, Technology and Society*, *19*, 5.

Matulionyte, R., & Lee, J.-A. (2022b). Copyright in AI-generated works: Lessons from recent developments in patent law. *SCRIPT-Ed*, *19*(1), 5–35. https://doi.org/10.2966/scrip.190122.5

McIntosh, T. R., Susnjak, T., Liu, T., Watters, P., Nowrozy, R., & Halgamuge, M. N. (2024). From COBIT to ISO 42001: Evaluating cybersecurity frameworks for opportunities, risks, and regulatory compliance in commercializing large language models. *Computers & Security*, *144*, 103964. https://doi.org/10.1016/j.cose.2024.103964

Methnani, L., Aler Tubella, A., Dignum, V., & Theodorou, A. (2021). Let me take over: Variable autonomy for meaningful human control. *Frontiers in Artificial Intelligence*, *4*. https://doi.org/10.3389/frai.2021.737072

Migliorini, S. (2024). China's interim measures on generative AI: Origin, content and significance. *Computer Law and Security Review*, *53*. Scopus. https://doi.org/10.1016/j.clsr.2024.105985

Milmo, D. (2023, June 23). Two US lawyers fined for submitting fake court citations from ChatGPT. *The Guardian*. https://www.theguardian.com/technology/2023/jun/23/two-us-lawyers-fined-submitting-fake-court-citations-chatgpt

Milne, S., Sheeran, P., & Orbell, S. (2000). Prediction and intervention in health-related behavior: A meta-analytic review of protection motivation theory.

Journal of Applied Social Psychology, *30*(1), 106–143. https://doi.org/10.1111/j.1559-1816.2000.tb02308.x

Mohsin, A., Janicke, H., Wood, A., Sarker, I. H., Maglaras, L., & Janjua, N. (2024). *Can we trust large language models generated code? A framework for in-context learning, security patterns, and code evaluations across diverse LLMs (No. arXiv:2406.12513). arXiv.* https://doi.org/10.48550/arXiv.2406.12513

Nahavandi, S. (2017). Trusted autonomy between humans and robots: Toward human-on-the-loop in robotics and autonomous systems. *IEEE Systems, Man, and Cybernetics Magazine*, *3*(1), 10–17. IEEE Systems, Man, and Cybernetics Magazine. https://doi.org/10.1109/MSMC.2016.2623867

Naik, K., & Tripathy, P. (2011). *Software testing and quality assurance: Theory and practice. John Wiley & Sons.*

Nicholas, G., & Friedl, P. (2024). *Regulating large language models: A roundtable report (No. arXiv:2403.15397). arXiv.* https://doi.org/10.48550/arXiv.2403.15397

Novelli, C., Taddeo, M., & Floridi, L. (2024). Accountability in artificial intelligence: What it is and how it works. *AI & Society, 39*(4), 1871–1882. https://doi.org/10.1007/s00146-023-01635-y

Ong, A. K. S., Prasetyo, Y. T., Tapiceria, R. P. K. M., Nadlifatin, R., & Gumasing, M. J. J. (2024). Factors affecting the intention to use COVID-19 contact tracing application "StaySafe PH": Integrating protection motivation theory, UTAUT2, and system usability theory. *PLOS ONE*, *19*(8), e0306701. https://doi.org/10.1371/journal.pone.0306701

Ozkaya, I. (2023). The next frontier in software development: AI-augmented software development processes. *IEEE Software, 40*(4), 4–9. https://doi.org/10.1109/MS.2023.3278056

Pan, R., Kim, M., Krishna, R., Pavuluri, R., & Sinha, S. (2024). *Multi-language unit test generation using LLMs* (No. arXiv:2409.03093). *arXiv*. https://doi.org/10.48550/arXiv.2409.03093

Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? assessing the security of GitHub Copilot's code contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*, 754–768. https://doi.org/10.1109/SP46214.2022.9833571

Peng, C., Yang, X., Chen, A., Smith, K. E., PourNejatian, N., Costa, A. B., Martin, C., Flores, M. G., Zhang, Y., Magoc, T., Lipori, G., Mitchell, D. A., Ospina, N. S., Ahmed, M. M., Hogan, W. R., Shenkman, E. A., Guo, Y., Bian, J., & Wu, Y. (2023). A study of generative large language model for medical research and healthcare. *NPJ Digital Medicine*, *6*(1), 1–10. https://doi.org/10.1038/s41746-023-00958-w

Peng, Q., Chai, Y., & Li, X. (2024). *HumanEval-XL: A multilingual code generation benchmark for cross-lingual natural language generalization* (No. arXiv:2402.16694). arXiv. https://doi.org/10.48550/arXiv.2402.16694

Rippetoe, P. A., & Rogers, R. W. (1987). Effects of components of protection-motivation theory on adaptive and maladaptive coping with a health threat. *Journal of Personality and Social Psychology*, *52*(3), 596–604. https://doi.org/10.1037/0022-3514.52.3.596

Rogers, R. W. (1975). A protection motivation theory of fear appeals and attitude change. *The Journal of Psychology*, *91*(1), 93–114. https://doi.org/10.1080/00223980.1975.9915803

Rogers, R. W., & Maddux, J. E. (1983). Protection motivation and self-efficacy: A revised theory of fear appeals and attitude change. *Journal of Experimental Social Psychology*, *19*(5), 469–479. https://doi.org/10.1016/0022-1031(83)90023-9

Roziere, B., Lachaux, M.-A., Chanussot, L., & Lample, G. (2020). Unsupervised translation of programming languages. *Advances in Neural Information Processing Systems*, *33*, 20601–20611. https://proceedings.neurips.cc/paper/2020/hash/ed23fbf18c2cd35f8c7f8de44f85c08d-Abstract.html

Ruiter, R. A. C., Kessels, L. T. E., Peters, G.-J. Y., & Kok, G. (2014). Sixty years of fear appeal research: Current state of the evidence. *International Journal of Psychology*, *49*(2), 63–70. https://doi.org/10.1002/ijop.12042

Şahin, E., Arslan, N. N., & Özdemir, D. (2025). Unlocking the black box: An in-depth review on interpretability, explainability, and reliability in deep

learning. *Neural Computing and Applications*, *37*(2), 859–965. https://doi.org/10.1007/s00521-024-10437-2

Shirafuji, A., Oda, Y., Suzuki, J., Morishita, M., & Watanobe, Y. (2023). Refactoring programs using large language models with few-shot examples. *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, 151–160. https://doi.org/10.1109/APSEC60848.2023.00025

Shypula, A., Madaan, A., Zeng, Y., Alon, U., Gardner, J., Hashemi, M., Neubig, G., Ranganathan, P., Bastani, O., & Yazdanbakhsh, A. (2024). *Learning performance-improving code edits* (No. arXiv:2302.07867). *arXiv.* https://doi.org/10.48550/arXiv.2302.07867

Slade, E. L., Dwivedi, Y. K., Piercy, N. C., & Williams, M. D. (2015). Modeling consumers' adoption intentions of remote mobile payments in the United Kingdom: Extending UTAUT with innovativeness, risk, and trust. *Psychology & Marketing*, *32*(8), 860–873. https://doi.org/10.1002/mar.20823

Sommerville, I. (2016). *Software engineering* (10th edition). Pearson.

Tamilmani, K., Rana, N. P., Wamba, S. F., & Dwivedi, R. (2021). The extended unified theory of acceptance and use of technology (UTAUT2): A systematic literature review and theory evaluation. *International Journal of Information Management*, *57*, 102269. https://doi.org/10.1016/j.ijinfomgt.2020.102269

Tang, Y., Liu, Z., Zhou, Z., & Luo, X. (2024). ChatGPT vs SBST: A comparative assessment of unit test suite generation. *IEEE Transactions on Software Engineering*, *50*(6), 1340–1359. IEEE Transactions on Software Engineering. https://doi.org/10.1109/TSE.2024.3382365

Thompson, N., McGill, T. J., & Wang, X. (2017). "Security begins at home": Determinants of home computer and mobile device security behavior. *Computers & Security*, *70*, 376–391. https://doi.org/10.1016/j.cose.2017.07.003

Tian, R., Ye, Y., Qin, Y., Cong, X., Lin, Y., Pan, Y., Wu, Y., Hui, H., Liu, W., Liu, Z., & Sun, M. (2024). *DebugBench: Evaluating debugging capability of*

large language models (No. arXiv:2401.04621). *arXiv*. https://doi.org/10.48550/arXiv.2401.04621

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). *LLaMA: Open and efficient foundation language models* (No. arXiv:2302.13971). *arXiv.* https://doi.org/10.48550/arXiv.2302.13971

Tsai, H. S., Jiang, M., Alhabash, S., LaRose, R., Rifon, N. J., & Cotten, S. R. (2016). Understanding online safety behaviors: A protection motivation theory perspective. *Computers & Security*, *59*, 138–150. https://doi.org/10.1016/j.cose.2016.02.009

Urbach, N., & Ahlemann, F. (2010). Structural equation modeling in information systems research using partial least squares. In *Journal of Information Technology Theory and Application, 11*(2), 5–40.

Vallance, C. (2023). *Sarah Silverman sues Meta and OpenAI, alleging they used her book without permission to train A.I. models.* People.com. https://people.com/sarah-silverman-sues-meta-openai-7559208

Varghese, J. R., & Thomas, D. S. (2024). Survey of generative AI in code generation: Privacy, security, and ethical considerations. *International Journal of Scientific Research*, *10*(4).

Vasani, S. (2024, September 25). *'Robot lawyer' company faces $193,000 fine as part of FTC's AI crackdown*. The Verge. https://www.theverge.com/2024/9/25/24254405/federal-trade-commission-donotpay-robot-lawyers-artificial-intelligence-scams

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention is all you need* (No. arXiv:1706.03762). *arXiv*. https://doi.org/10.48550/arXiv.1706.03762

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS Quarterly*, *27*(3), 425. https://doi.org/10.2307/30036540

Venkatesh, V., Thong, J. Y. L., & Xu, X. (2012). Consumer acceptance and use of information technology: Extending the unified theory of acceptance and use of technology. *MIS Quarterly*, *36*(1), 157. https://doi.org/10.2307/41410412

Venkatesh, V. (2022). Adoption and use of AI tools: A research agenda grounded in UTAUT. *Annals of Operations Research*, *308*(1), 641–652. https://doi.org/10.1007/s10479-020-03918-9

Verkijika, S. F. (2018). Understanding smartphone security behaviors: An extension of the protection motivation theory with anticipated regret. *Computers & Security*, *77*, 860–870. https://doi.org/10.1016/j.cose.2018.03.008

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Singer, Y., & Shieber, S. (2020). Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, *33*, 12388–12401. https://proceedings.neurips.cc/paper/2020/hash/92650b2e92217715fe312e6fa7b90d82-Abstract.html

Vimalkumar, M., Sharma, S. K., Singh, J. B., & Dwivedi, Y. K. (2021). 'Okay google, what about my privacy?': User's privacy perceptions and acceptance of voice based digital assistants. *Computers in Human Behavior*, *120*, 106763. https://doi.org/10.1016/j.chb.2021.106763

Wadhwa, N., Pradhan, J., Sonwane, A., Sahu, S. P., Natarajan, N., Kanade, A., Parthasarathy, S., & Rajamani, S. (2023). *Frustrated with code quality issues? LLMs can help!* (No. arXiv:2309.12938). *arXiv*. https://doi.org/10.48550/arXiv.2309.12938

Wang, J., & Chen, Y. (2023). A review on code generation with LLMs: Application and evaluation. In *2023 IEEE International Conference on Medical Artificial Intelligence (MedAI)*, 284–289. https://doi.org/10.1109/MedAI59581.2023.00044

Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). *Software testing with large language models: Survey, landscape, and vision* (No. arXiv:2307.07221). arXiv. https://doi.org/10.48550/arXiv.2307.07221

Wang, R., Cheng, R., Ford, D., & Zimmermann, T. (2024). Investigating and designing for trust in AI-powered code generation tools. *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, 1475–1493. https://doi.org/10.1145/3630106.3658984

Weisz, J. D., Kumar, S., Muller, M., Browne, K.-E., Goldberg, A., Heintze, E., & Bajpai, S. (2024). *Examining the use and impact of an AI code assistant on developer productivity and experience in the enterprise* (No. arXiv:2412.06603; Version 1). *arXiv*. https://doi.org/10.48550/arXiv.2412.06603

Werts, C. E., Linn, R. L., & Jöreskog, K. G. (1974). Intraclass reliability estimates: Testing structural assumptions. *Educational and Psychological Measurement*, *34*(1), 25–33. https://doi.org/10.1177/001316447403400104

Wiest, I. C., Ferber, D., Zhu, J., van Treeck, M., Meyer, S. K., Juglan, R., Carrero, Z. I., Paech, D., Kleesiek, J., Ebert, M. P., Truhn, D., & Kather, J. N. (2024). Privacy-preserving large language models for structured medical information retrieval. *NPJ Digital Medicine*, *7*(1), 1–9. https://doi.org/10.1038/s41746-024-01233-2

Williams, M. D., Rana, N. P., & Dwivedi, Y. K. (2015). The unified theory of acceptance and use of technology (UTAUT): A literature review. *Journal of Enterprise Information Management*, *28*(3), 443–488. https://doi.org/10.1108/JEIM-09-2014-0088

Wu, T., Luo, L., Li, Y.-F., Pan, S., Vu, T.-T., & Haffari, G. (2024). *Continual learning for large language models: A Survey* (No. arXiv:2402.01364). *arXiv*. https://doi.org/10.48550/arXiv.2402.01364

Xin, T., Siponen, M., & Chen, S. (2021). Understanding the inward emotion-focused coping strategies of individual users in response to mobile malware threats. *Behaviour & Information Technology*, *41*(13), 2835–2859. https://doi.org/10.1080/0144929X.2021.1954242

Xu, W., Gao, K., He, H., & Zhou, M. (2024). *LiCoEval: Evaluating LLMs on license compliance in code generation* (No. arXiv:2408.02487). *arXiv*. https://doi.org/10.48550/arXiv.2408.02487

Yazdanmehr, A., Li, Y., & Wang, J. (2023). Employee responses to information security related stress: Coping and violation intention. *Information Systems Journal*, *33*(3), 598–639. https://doi.org/10.1111/isj.12417

Zampano, G. (2024, December 20). *Italy's privacy watchdog fines OpenAI for ChatGPT's violations in collecting users' personal data*. AP News. https://apnews.com/article/italy-privacy-authority-openai-chatgpt-fine-6760575ae7a29a1dd22cc666f49e605f

Zhang, K., Li, Z., Li, J., Li, G., & Jin, Z. (2023). *Self-Edit: Fault-aware code editor for code generation* (No. arXiv:2305.04087). arXiv. https://doi.org/10.48550/arXiv.2305.04087

Zhang, K., Wang, D., Xia, J., Wang, W. Y., & Li, L. (2023). ALGO: Synthesizing algorithmic programs with generated oracle verifiers. *Advances in Neural Information Processing Systems*, *36*, 54769–54784.

Zhang, L., & McDowell, W. C. (2009). Am I really at risk? Determinants of online users' intentions to use strong passwords. *Journal of Internet Commerce*, *8*(3–4), 180–197. https://doi.org/10.1080/15332860903467508

Zhao, W., Ren, X., Hessel, J., Cardie, C., Choi, Y., & Deng, Y. (2024). *WildChat: 1M ChatGPT interaction logs in the wild* (No. arXiv:2405.01470). arXiv. https://doi.org/10.48550/arXiv.2405.01470